Universidad de Oviedo

Manual del programador del Trabajo Fin de Máster realizado por

IKER PLAZAOLA ORMAZABAL

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Análisis Arquitectura HW y Plataformas de Programación

FECHA DE PRESENTACIÓN: Enero del 2015

ÍNDICE

ÍNDICE DE TABLAS

1 DESCRIPCIÓN DEL DOCUMENTO

El principal fin de este documento es el de mostrar al programador todo lo desarrollado en este proyecto. De esta manera, cualquier programador podrá continuar desde el punto en que se dejó el proyecto.

2 RECURSOS EMPLEADOS

2.1 RECUSOS HARDWARE

Para un correcto funcionamiento del STEP 7 Basic / Professional V13, también conocido como TIA Portal, su PC debe cumplir como mínimo con los siguientes requisitos:

- Procesador: CoreTM i5-3320M 3,3 GHz o similar
- Memoria de trabajo: 8 GB (recomendado) o más
- Disco duro: 300 GB SSD
- Tarjeta gráfica: Mínimo 1920 x 1080
- Monitor: 15,6" Wide Screen Display (1920 x 1080)

2.2 RECUSOS SOFTWARE

Para que todos los programas funcionen de una manera adecuada, es imprescindible que el portátil tenga instalado los siguientes softwares:

- Sistema operativo MS Windows 7 Professional + SP1
- Programa de Totally Integrated Automation Portal V13
- SIMATIC STEP 7 Professional V13 + STEP 7 Safety V13 + WinCC BASIC V13
- Sysmac studio de OMRON versión 1.0.9.47
- TwinCAT XAE Base de Beckhoff versión 3.1.0.0 + TwinCAT XAE PLC 3.1.0.0
- CODESYS V3.5 SP4 patch 4

2.3 RECUSOS HUMANOS

La persona encargada del desarrollo de este proyecto seria aconsejable que tuviese conocimientos mínimos en:

- Estándar de programación de ULMA
- Dominio de la plataforma de SIEMENS STEP7
- Conocimientos básicos en TIA PORTAL
- Conocimiento mínimo de la normativa IEC 61131 y sus en variantes como es la Programación Orientada a Objeto.

3 DESARROLLO DE LA SOLUCIÓN

3.1.-PROYECTO EN CODESYS

3.1.1-COMO CREAR UN PROYECTO EN CODESYS

A continuación se detallaran los pasos para programar con la programación orientada a objeto en Codesys. La última versión de Codesys lo podrá encontrar en esta dirección http://www.codesys.com/download.html). En este caso se instaló la siguiente versión:

Ilustración 1: Versión Codesys

Una vez instalado, lo próximo es crear un nuevo proyecto. El programa ofrece tres opciones: proyecto vacío, proyecto estándar y proyecto estándar con compositor de aplicaciones. Para una prueba simple se recomienda escoger la opción del proyecto estándar, y a continuación se abrirá la siguiente ventana:

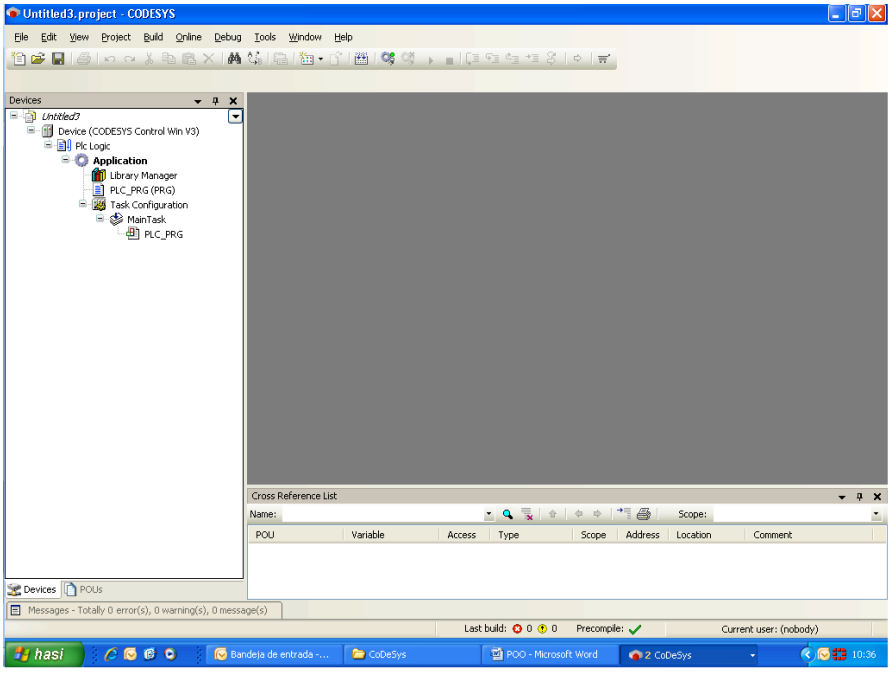


Ilustración 2: Ventana Principal

Como esta técnica consiste en definir objetos con sus atributos y métodos, fue necesario realizar un estudio de las funciones para así definir que es un objeto y cuáles serán sus métodos y atributos.

La primera función de mesa escogida fue la FC420 (ZPA_tarjeta_CB), esta es una función para una zona donde exista una tarjeta CB y se quiera hacer una acumulación sin presión. En cada uno de los tramos solo entra una caja.

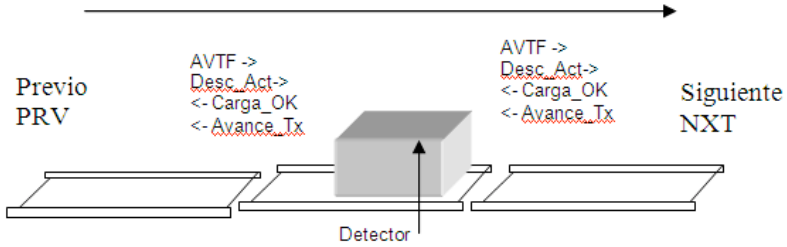


Ilustración 3: Señales de transferencia FC420

Y la segunda función será la FC425 (TL_Sep_Salida), la principal funcionalidad de esta función es la de asegurar que la salida de cargas se realice con una secuencia mínima determinada. En este caso en cada tramo puede haber más de una caja.

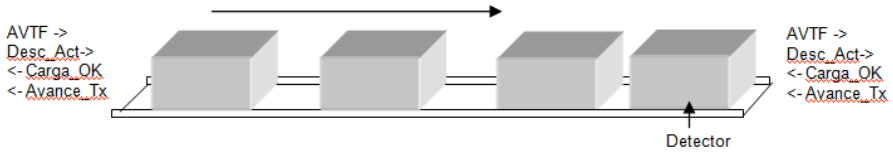


Ilustración 4: Señales de transferencia FC425

Por lo tanto, la idea principal es que como las dos funciones tienen parecida funcionalidad puedan compartir los atributos y métodos. Los métodos que tendrán en común cada mesa serán los siguientes:

Lleno→ Este método se encarga de enviar el aviso de transferencia a la siguiente mesa. Este método es llamado cuando la mesa se encuentra cargada.

Parar→ Este método es el encargado de parar el avance del transportador siempre y cuando se de alguna de las casuísticas programadas.

Vaciando→ Es el encargado de activar el bit de descarga siempre y cuando la mesa siguiente le dé la orden de poder descargar.

Vaciar/Parar→ En este caso activaremos el bit de avance para que la mesa pueda ser cargada o descargada.

Vacio→ Este método es el encargado de enviar a la mesa anterior que se encuentra vacía y a disposición de ser cargada.

Y los atributos que compartirán serán Estadozona y Estadomesa. Como sus nombres indican, Estadozona nos avisara sobre el estado una zona de la instalación (si está en marcha, paro, reset, etc.). Y Estadomesa nos indicará el estado de cada mesa (si ha habido algún error, el estado en el que se encuentran las señales de entrada y salida, la mesa en que etapa se encuentra (etapa de carga, etapa de descarga, cargado, vaciado), etc).

3.1.2-COMO PROGRAMAR CON PROGRAMACIÓN ORIENTADA A OBJETO

Una vez definidos los atributos y métodos que conformaran las dos funciones, comenzaremos con la estructura que tendrá el programa en la nueva plataforma con la nueva técnica de programación. De ahora en adelante cada función se llamara clase, como es un lenguaje de alto nivel y está basado en "C" las terminologías cambian puesto que con esta técnica además de los electrónicos los informáticos también estarán cualificados para modificar el programa.

PASO 1:

Crearemos las dos clases dentro de la aplicación que se creó automáticamente cuando se escogió un proyecto estándar. Para ello pulsara botón derecho sobre aplicación, escogiendo añadir objeto y después seleccionara POU (Program Organization Unit).

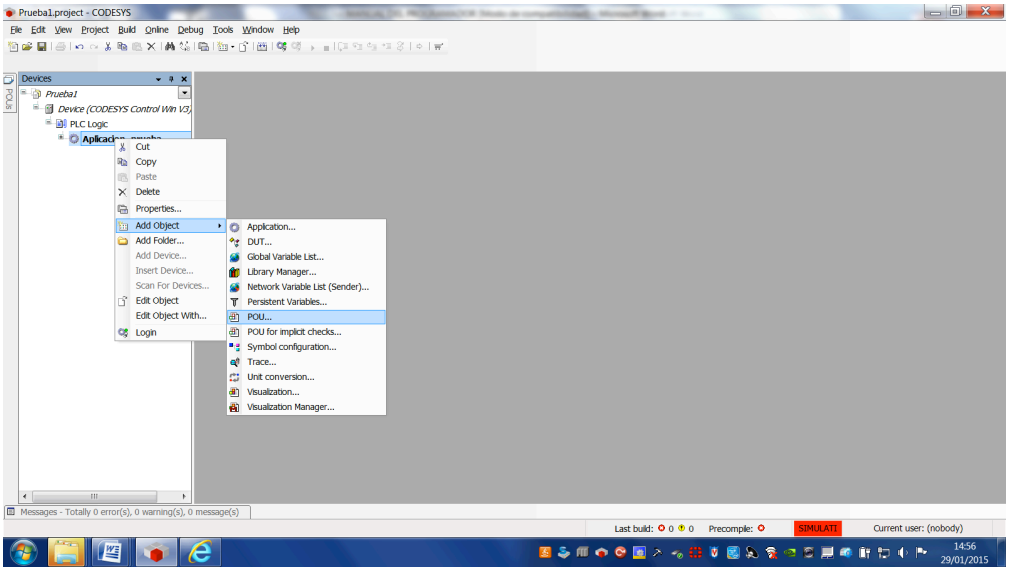


Ilustración 5: Como crear una función

Una vez pulsado sobre POU nos dará la opción de escoger el tipo de función que se quiera crear.

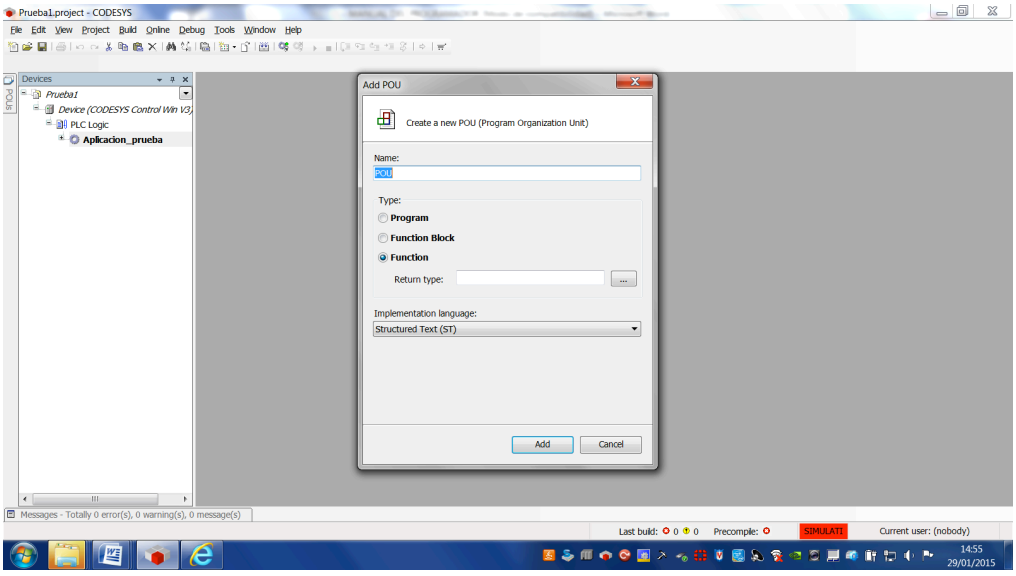


Ilustración 6: Selección del tipo de función

PASO 2:

Una vez creadas las dos clases el siguiente paso que se dio fue el de programar las dos funciones con sus dos códigos. Por lo tanto, primero se programó la clase FC_420, con la idea de programar todos los métodos del primer objeto o mesa y así después poder compartir estos métodos con otras funciones.

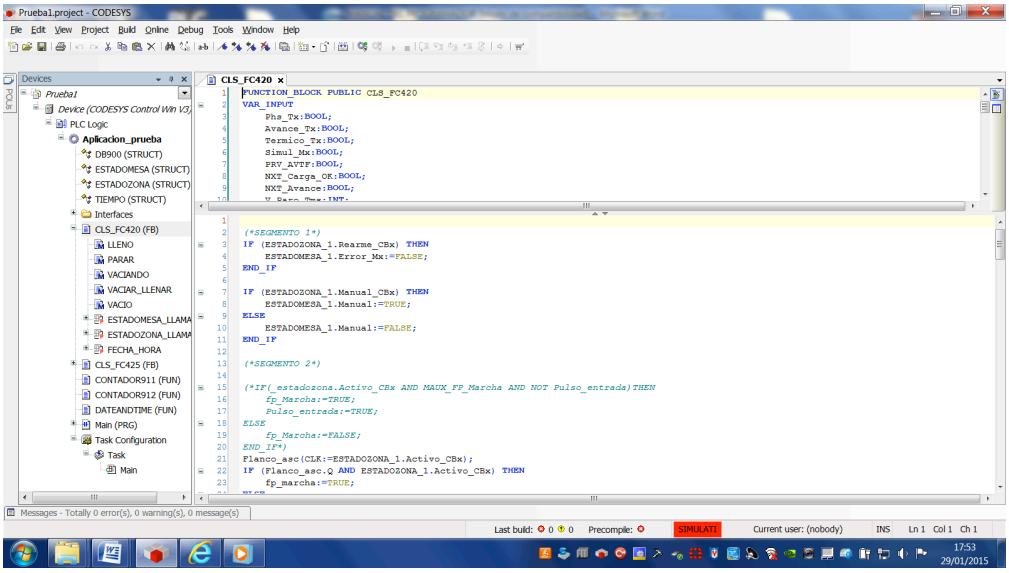


Ilustración 7: Crear la clase FC420 con sus métodos

PASO 3:

El siguiente paso fue el de programar la clase FC_425 a su vez compartiendo los métodos de otras funciones. Pero no fue posible, ya que, aunque eran parecidas las funcionalidades de cada función, realmente cada una tenía condiciones diferentes para realizar la acción, por lo tanto, se tuvo que crear métodos para cada función perdiendo así por completo la gran ventaja de programar con Programación Orientada a Objeto.

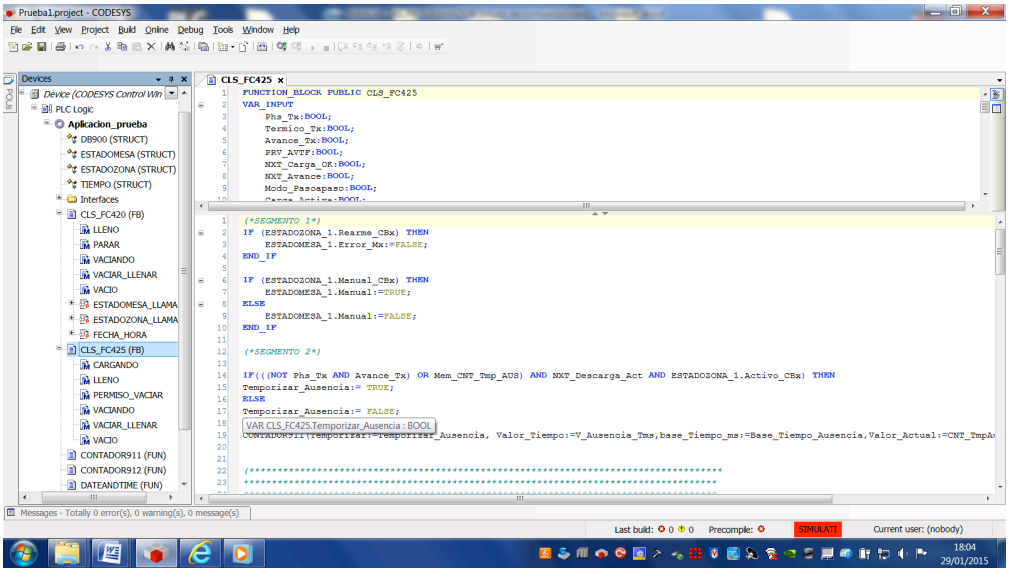


Ilustración 8: Crear la clase FC425 con sus métodos

Aunque se hizo una simulación donde la función FC425 compartía el método de lleno de la función FC420.

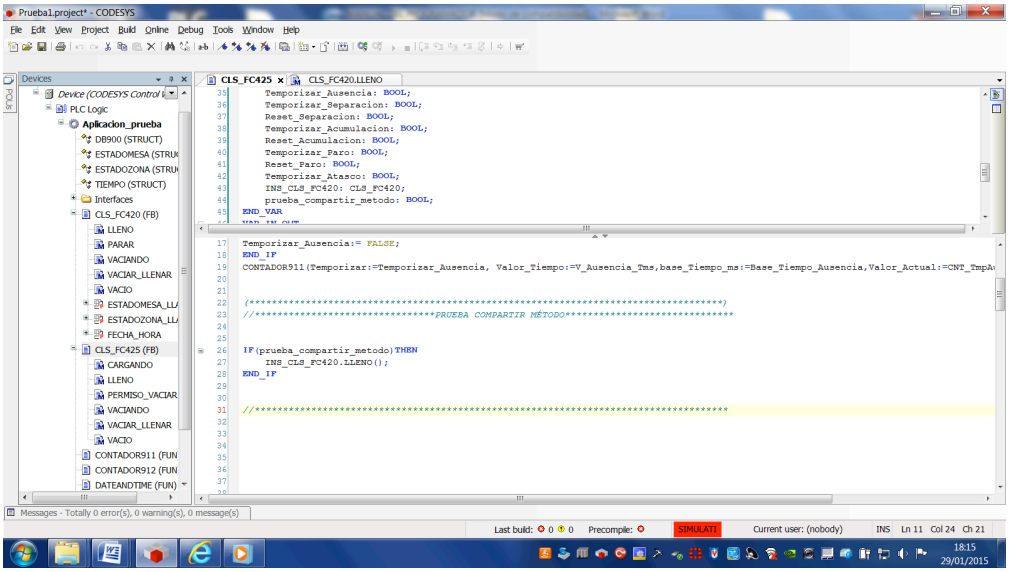


Ilustración 9: Compartir método entre dos funciones

A continuación un claro ejemplo de porque no se podían compartir los métodos, en la imagen de la izquierda se puede ver que las condiciones para vaciar o llenar son totalmente diferentes a los de la derecha.

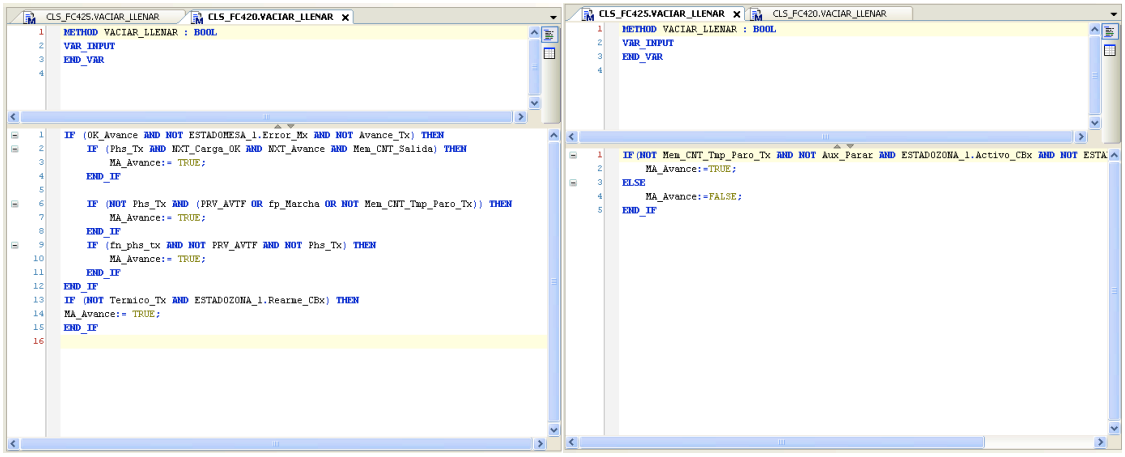


Ilustración 10: Diferentes condiciones para activar Marca Avance

PASO 4:

Para finalizar con la programación de las dos funciones con Programación Orientada a Objeto, quedaba solamente crear la función para la obtención de la hora pero he de decir que la programación de esta función fue más compleja de lo previsto, hasta tal punto que se decidio dejarlo sin programar puesto que no se consiguió que compilase sin errores.

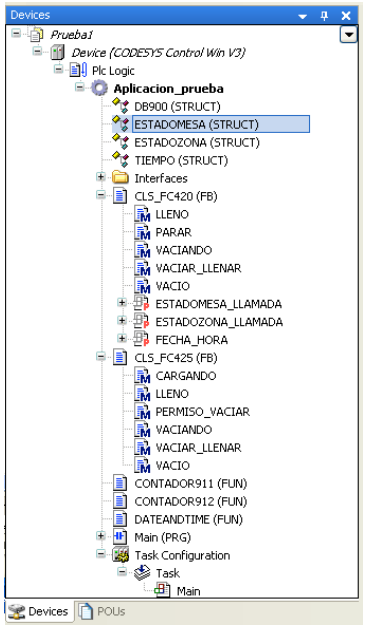


Ilustración 11: Árbol del programa

Llegados a este punto y visto el éxito cosechado, se decidió cambiar de estrategia y programar en otra plataforma que cumpliese con el estándar IEC61131 dejando a un lado la plataforma de codesys y la Programación Orientada a Objeto.

3.2 MIGRACIÓN STEP7 (Programado en Lenguaje de Contactos) a TIA PORTAL (Programado en Texto Estructurado)

La siguiente fase de mi proyecto, consistió en migrar un proyecto previamente diseñado e implantado en Step 7 a Tia Portal. Y a su vez se aprovecho para traducir a un lenguaje de alto nivel como es el texto estructurado.

El proyecto escogido para dicha migración es un proyecto pequeño, puesto que lo demás, tendríamos un programa muy complejo para su traducción y simulación. Por lo tanto, el proyecto escogido es una zona del proyecto de Aeropuertos de Confins, en concreto, la parte de llegadas del aeropuerto. Esta zona al estar compuesta de tres hipódromos de mismas dimensiones y funcionalidades, con que solamente se programe una de ellas nos será suficiente. En la siguiente imagen podéis observar como es la zona de llegadas de la instalación.

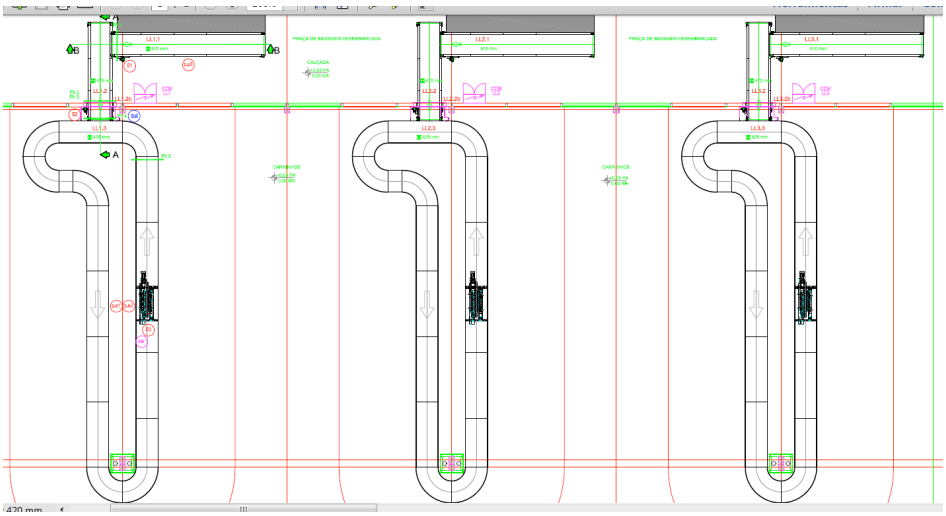


Ilustración 12: Zona llegadas Confins

3.2.1-COMO MIGRAR DE UNA PLATAFORMA A OTRA

Paso 1: Instalación correcta de los dos programas

Se deben tener instalados todos los programas con las versiones adecuadas. Para ello véase el **manual del instalador**.

Paso 2: Migrar un programa

Una vez abierto el TIA Portal aparecerá la siguiente ventana y clicarán en el apartado Migrar proyecto.

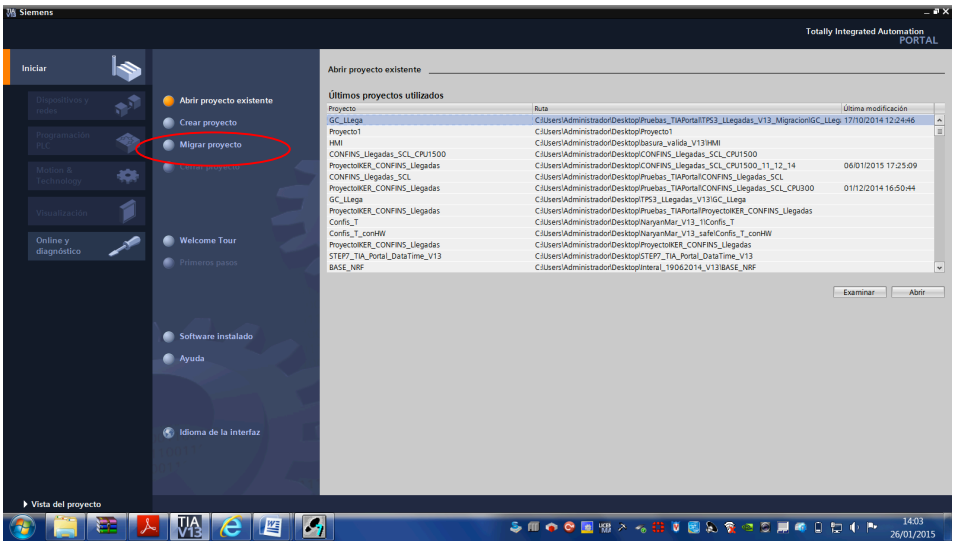


Ilustración 13: Sección para migrar de Step7 a TIA Portal

Según se pulse en Migrar aparecerá la siguiente pantalla, en el cual habrá que especificar las diferentes rutas del programa; Ruta origen del programa, en este apartado indicaremos la ruta donde se encuentra el programa a traducir.

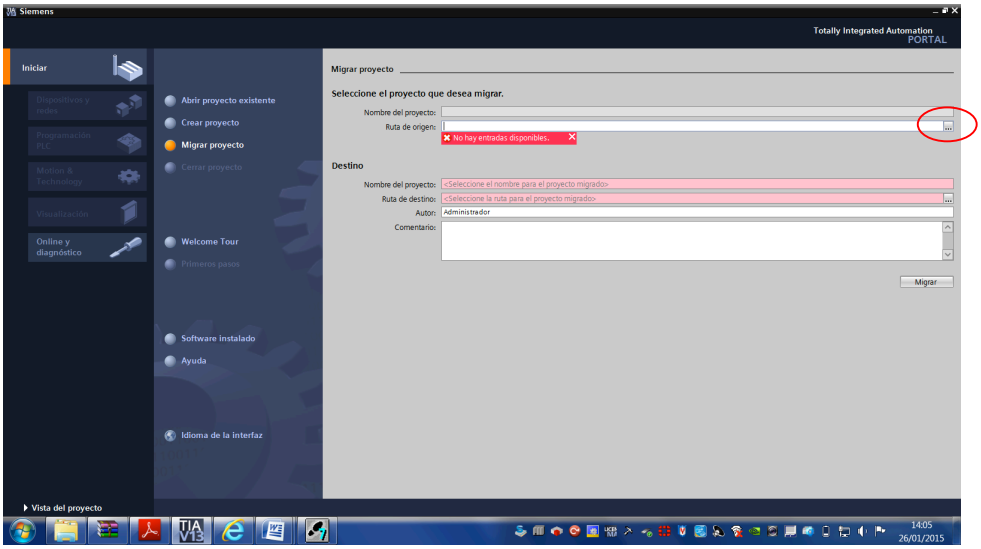


Ilustración 14: Insertar ruta de origen

El siguiente paso será el de encontrar donde está el archivo que después será migrado. La extensión del archivo será .s7p.

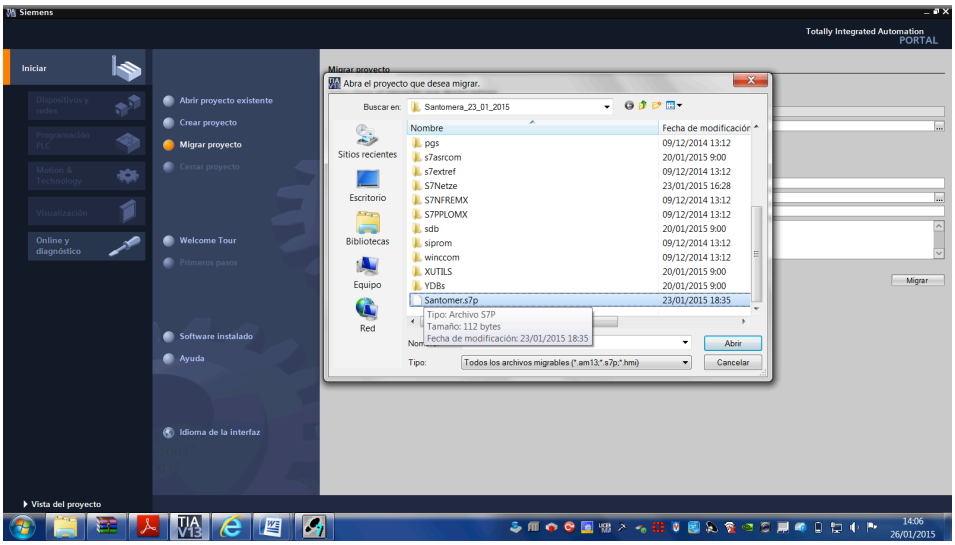


Ilustración 15: Seleccionar archivo

Y el programa automáticamente seleccionará una ruta de destino, en caso de no ser de su agrado seleccionará la ruta que más le convenga.

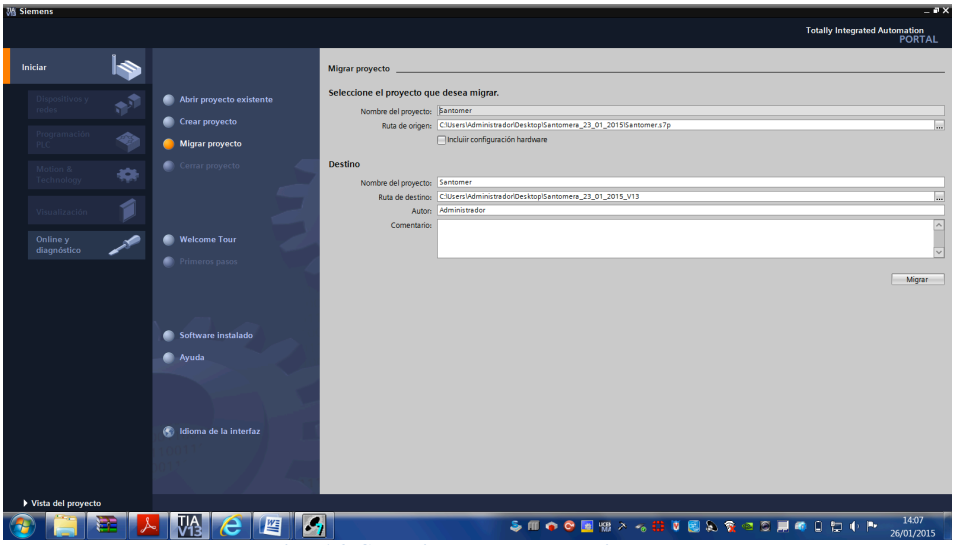


Ilustración 16: Seleccionar ruta de destino

Una vez seleccionadas las dos rutas y pulsado el icono de migrar, el programa automaticamente comenzará con la migración:

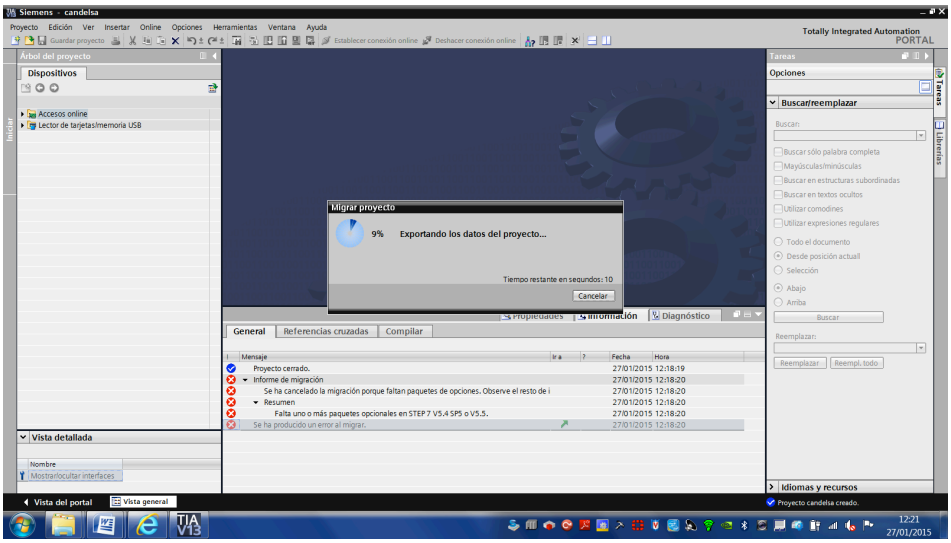


Ilustración 17: Imagen migrando el proyecto

Como es probable que el programa migrado por temas de incompatibilidad genere errores, el siguiente paso será depurar el programa hasta conseguir que la compilación sea sin errores.

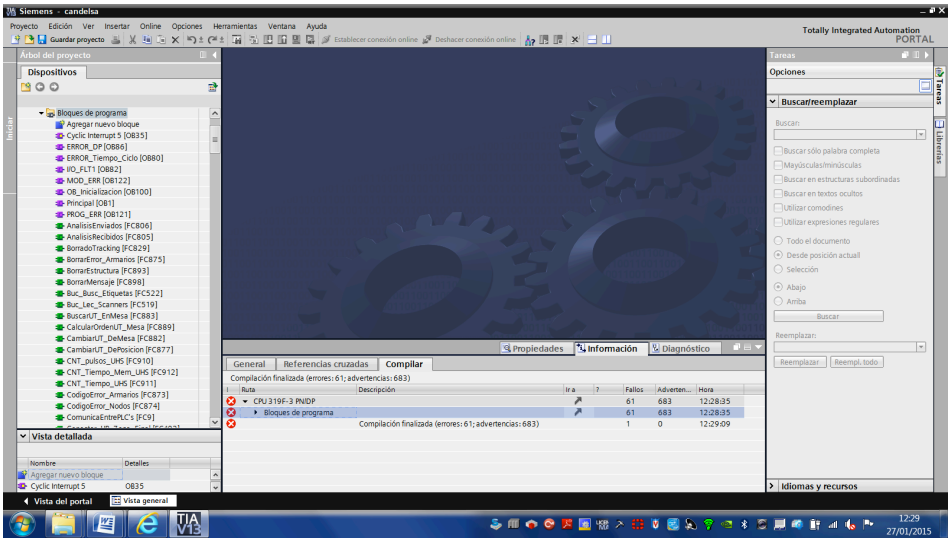


Ilustración 18: Estado del proyecto migrado

Paso 3: Compilar sin errores el programa

Una vez tengamos el programa cargado en el TIA Portal automáticamente se genera un árbol con los diferentes dispositivos y programas. Aunque la nueva estructura jerárquica cambia algo de la anterior verán que trabajar con ella es mucho más sencillo de lo que parece:

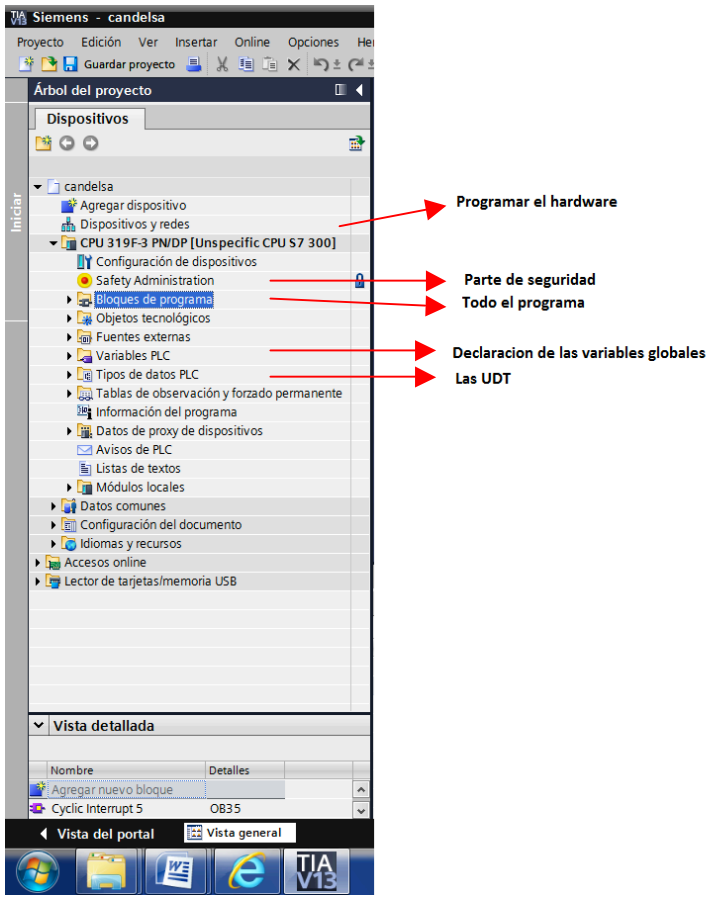


Ilustración 19: Árbol del proyecto

En esta nueva plataforma, clicando sobre el árbol con el botón derecho, nos aparecerá un desplegable con todas las funciones que permite el TIA Portal. Por ejemplo: Compilar, Cargar, insertar más funciones, comparar bloques, etc.

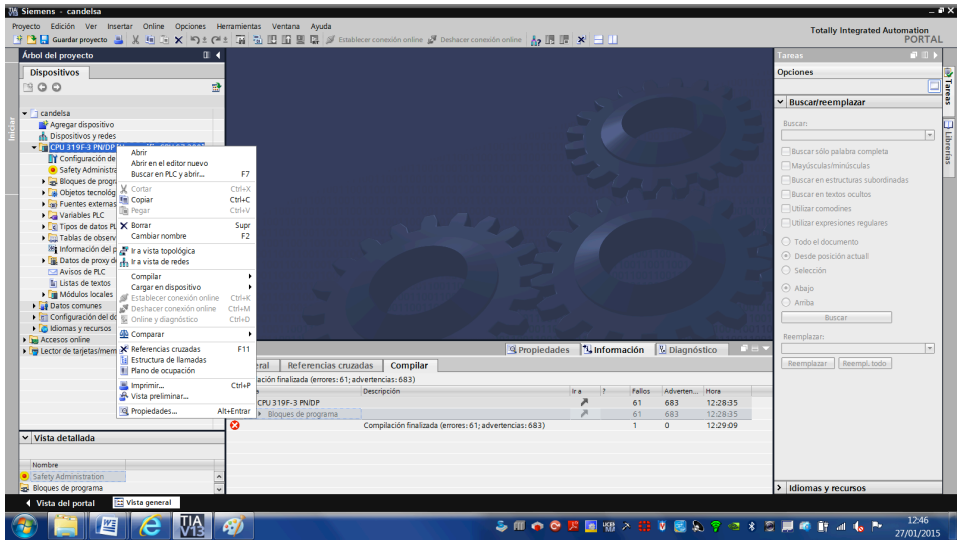


Ilustración 20: Como compilar un programa

Y como en este caso se tienen errores por incompatibilidades, se clicará sobre la opción de compilar que este a su vez ofrece la opción de escoger que se quiere compilar. En este caso para que la compilación sea completa escogeremos siempre compilar todos los bloques.

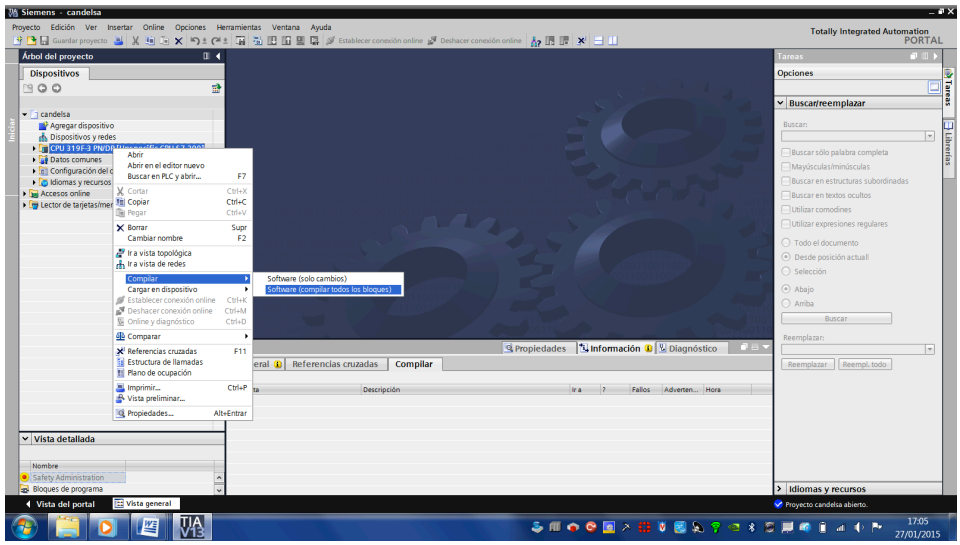


Ilustración 21: Compilar todo el software

Hasta que no se tenga todo compilado y sin errores, no avanzará hasta el siguiente paso. Al final quedándole algo del estilo de la siguiente imagen:

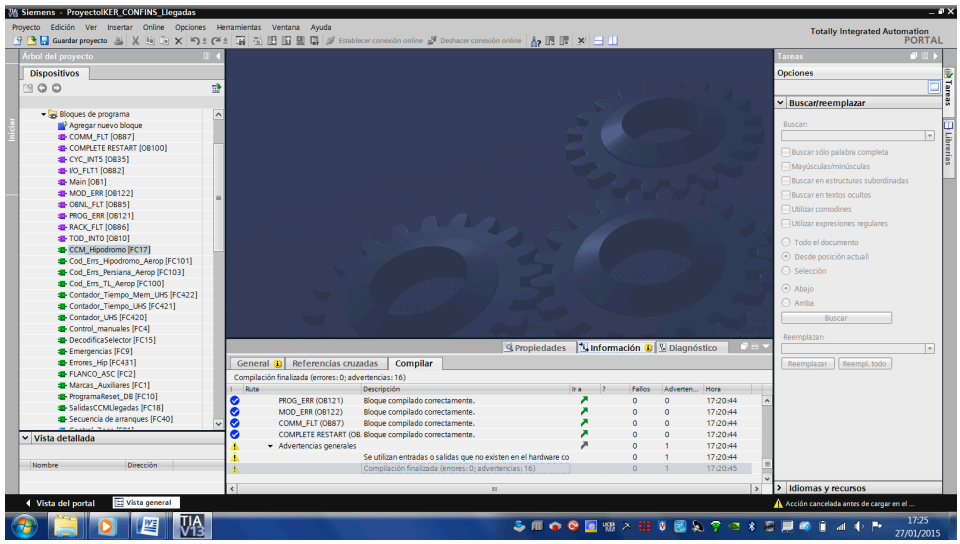


Ilustración 22: Proyecto compilado

Una vez migrado el programa a TIA Portal y compilado sin errores, solamente quedó traducirlo a un lenguaje de alto nivel que fuera exportable a otras plataformas. Por lo tanto como se menciono en la memoria se tradujo este programa a SCL, lenguaje equivalente a ST en Siemens, este es un lenguaje que cumple con la normativa IEC 61131-3.

3.2.2-TRADUCCION DEL PROGRAMA A SCL

3.2.2.1.-MAIN (OB1)

El programa Main, o también conocido como OB1, está dividido en 6 segmentos, cada uno de estos segmentos se encarga de una funcionalidad de la zona de llegadas del aeropuerto.

En la siguiente lista se detallan cada una de ellas:

a. **Marcas Auxiliares**: Este FC es el encargado de definir cada una de las marcas auxiliares que después a lo largo del programa se utilizarán. Como pueden ser el ON y OFF o los pulsos cada 0.1s, 0.2s, 0.5s, etc.

b. **Control Manuales**: En este FC se memorizaran todos los movimientos manuales.

c. **CCM Hipódromo**: Este es el FC donde controlaremos todos los movimientos de la zona de llegadas. Es aquí donde se activaran todas las marcas para así después poder activar las salidas correspondientes.

d. **Salidas CCM Llegadas**: Como su nombre indica en este FC activaremos las salidas dependiendo del estado de las marcas del FC anterior.

e. **Secuencia de arranques**: En este caso aquí se resetearan los DBs de Persianas y Hipódromos para que al arrancar no guarden valores anteriores.

f. **Errores Hipódromos**: Este FC será el encargado de escribir en el código de error cada vez que ocurra un error.

Y finalmente también se crearon diferentes carpetas con la finalidad de agrupar aquellas funciones que eran parecidas.

a) **Carpeta de contadores tiempo**, esta carpeta estará compuesta por dos funciones con la misma finalidad; calcular el tiempo transcurrido.

b) **Carpeta Flanco ascendente**, en esta carpeta estará la función para hacer un flanco ascendente, ya que, no hay otra opción más directa que esta. Además con esta medida se evitan posibles errores al migrar de una plataforma a otra.

c) **Carpeta Llamadas errores hipódromo**, dentro de esta carpeta están las funciones que después serán llamadas por la función de errores dentro del OB1.

d) **Carpeta Llamadas CCM Hipódromos**, dentro de esta carpeta están todos los bloques funcionales para el manejo de la zona de llegadas del aeropuerto.

e) **Carpeta obtener hora**, en esta carpeta está la función que calcula el tiempo que lleva en marcha la instalación.

f) **Carpeta reset DB**, mediante la función almacenada en esta carpeta se resetea cualquier DB.

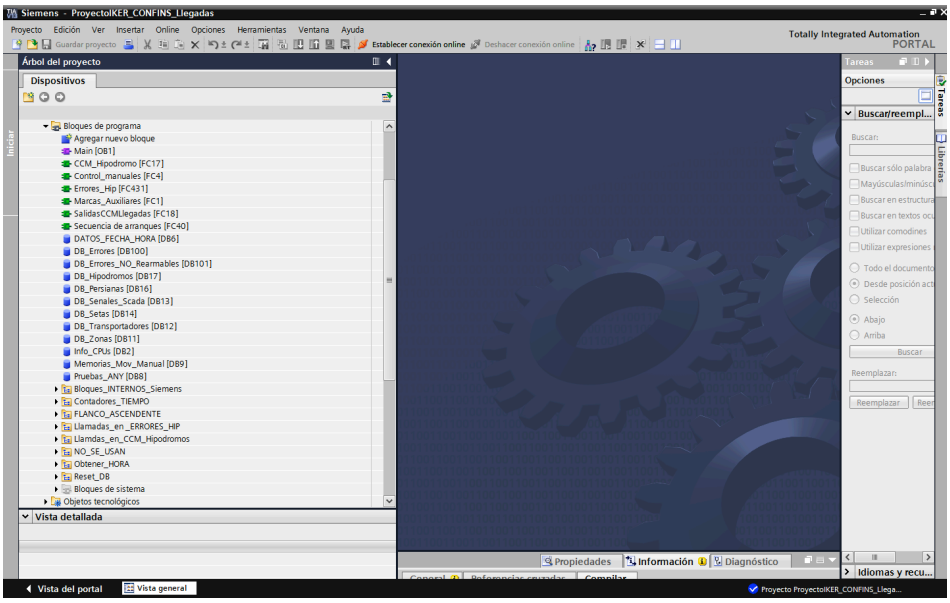


Ilustración 23: Estructura jerárquica del programa

3.2.2.2.-MARCAS AUXILIARES (FC1)

Aprovechando las marcas de ciclo que tiene Siemens se crearan diferentes pulsos a diferentes frecuencias, creando flancos positivos cada 0,1s 0,2s 0,5s 0,8s 1s 2s. Y también se calculara el tiempo transcurrido cada vez que se llame a esta función, esto sirve especialmente para evitar los retrasos en los contadores de tiempo.

Para la detección del flanco ascendente se creó una función, la función quedó de la siguiente manera:

Ilustración 24: Función Flanco Ascendente

3.2.2.3.-CONTROL MANUALES (FC4)

Esta función decodificará el número que el display marca, así accediendo al estado de esa mesa y cambiando su estado por el contrario. Esto es, si esta la mesa parada, el array tendrá un cero en esa posición, por lo tanto, lo pondremos a uno y si fuese al revés lo pondremos a cero.

3.2.2.4.-CCM HIPODROMOS (FC17)

Esta es la función encargada de gobernar todo la zona de llegadas del aeropuerto, por lo tanto, esta función además de llamar a diferentes funciones para la movimentación de las cintas activará todas las marcas emergencias que fuesen necesarias (balizas, sirenas, etc.).

a) Bloque funcional CONTROL_ZONA

Con esta función se controla el estado de la instalación, bien si está en marcha o paro y bien si está en error.

b) Bloque funcional TL SEP SALIDA FB LLEGADAS

Esta es la función encargada de activar y desactivar la primera cinta del aeropuerto, como todas las cintas tienen diferentes temporizadores para que estas entren en ahorro de energía en el caso de que no hubiese maletas en la cinta.

c) Bloque funcional TL INYECTOR HIP V2

Esta función activará y desactivará la segunda cinta. Pero ira relacionada con la cinta del hipódromo y la puerta, esto es, si la puerta está cerrada por mucho que el hipódromo le indique que está en marcha y puede introducir maletas, esta no podrá introducirlas hasta que la puerta se abra.

d) Bloque funcional LL1_HIPODROMO

Mediante esta función se activará la cinta del hipódromo donde los pasajeros podrán recoger sus maletas esta cinta tendrá dos sensores y pasado el tiempo del contador sin que esta fotocélula haya visto alguna maleta, este se parará.

e) Bloque funcional LL PERSIANA 1

Y por último esta la función encargada de subir o bajar la persiana dependiendo de las condiciones que se cumplan.

A continuación se enumeraran las diferentes modificaciones que se tuvieron que hacer dentro de la función de CCM_Hipodromo.

CASO 1: Instanciación de FB

Como todas estas funciones son bloques funcionales, FB, cada vez que se quieran llamarlas antes se deben instanciarlas. Por lo tanto, cuando se hace la llamada al bloque funcional en vez de mostrar el nombre de la función invocada escribirá el nombre del DB instanciado y dentro de paréntesis todas las entradas y salidas de esta función de bloque.

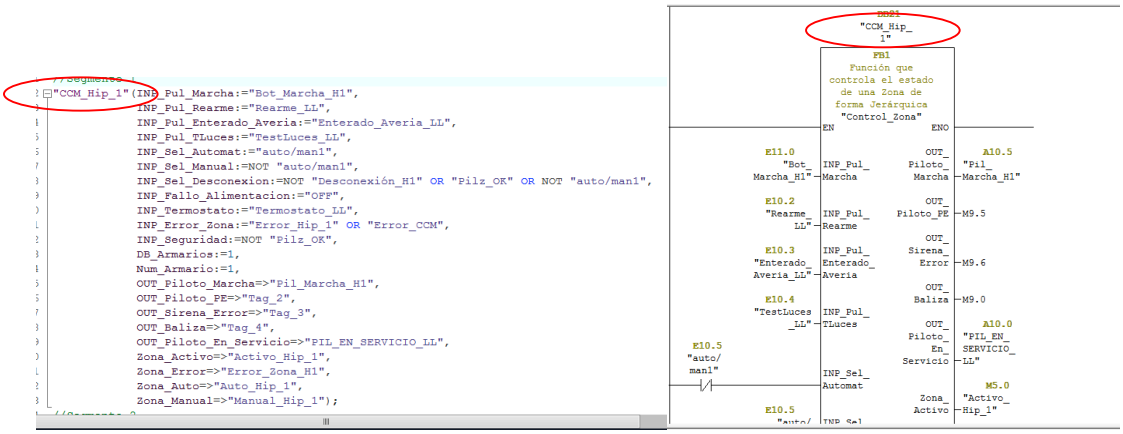


Ilustración 25: Instanciación de FB

CASO 2: Llamadas a variables declarados en DB

Otro caso es el de escritura en un DB concreto, como por ejemplo, cuando se detecta que la fotocélula del tope de arriba de la persiana esta accionada, debemos escribir en el DB_Persianas en el apartado de PersianaAbierta un "TRUE" o un "1", lo demás por defecto se mantendrá a "FALSE" o "0".

La manera de escribir esta condición en SCL sería la siguiente:

"DB_Persianas".Persianas[1].PersianaAbierta:=TRUE

Nombre del DB — *Posición del Array del tipo TRANS_Persiana que es una UDT* — *Nombre de la variable que queremos escribir*

La posición del Array en este caso es constante pero puede ser una variable que vaya en incremento o en decremento como se quiera.

```
FOR #i := 1 TO 200 BY 1
DO "DB_Errores".Zona[1].Err_Transp[#i] := FALSE;
END_FOR;
```

Ilustración 26: Ejemplo de cómo recorrer las variables de una estructura

CASO 3: Llamadas indirectas a DB

Otra opción que también se tuvo que traducir fue las llamadas indirectas a DBs o estructuras. Con las llamadas indirectas lo que se busca es minimizar las modificaciones en los bloques funcionales. En la siguiente imagen se ve reflejado mejor lo que se quiere hacer con estas llamadas:

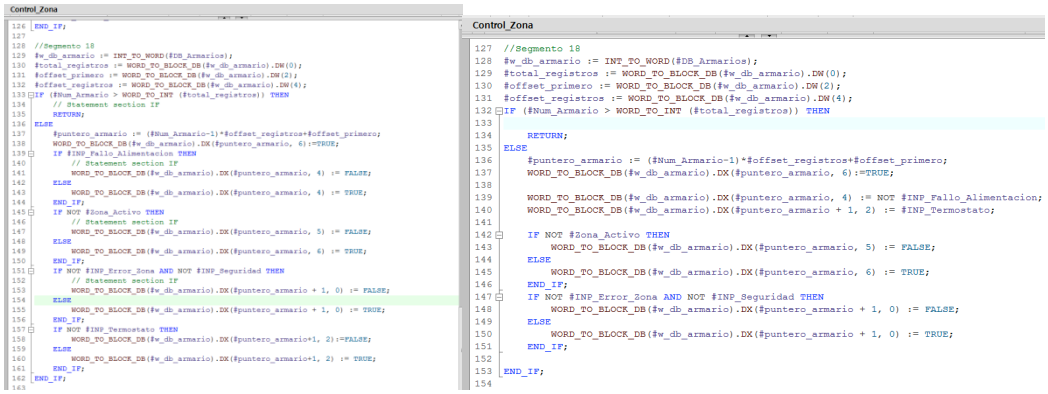


Ilustración 27: Llamada a DBs de forma indirecta

En la primera imagen se puede visualizar como sería la traducción literal del código y en la segunda se optimizaron aquellas condiciones que en texto estructura no tenían sentido y ralentizaban el programa. De esta manera el DB a abrir será aquel DB que corresponda con el número que se le asocie a DB_Armarios.

CASO 4: Variable IN/OUT del tipo estructura

Declarar como IN/OUT una variable que es del Tipo TRANS_Banda y cuando hacemos la llamada a la función de bloque donde será utilizada, se le asignara la dirección del DB correspondiente. De esta manera escribiendo el nombre de la declaración escribiremos en el DB seleccionado. En el siguiente ejemplo lo veremos con mayor claridad:

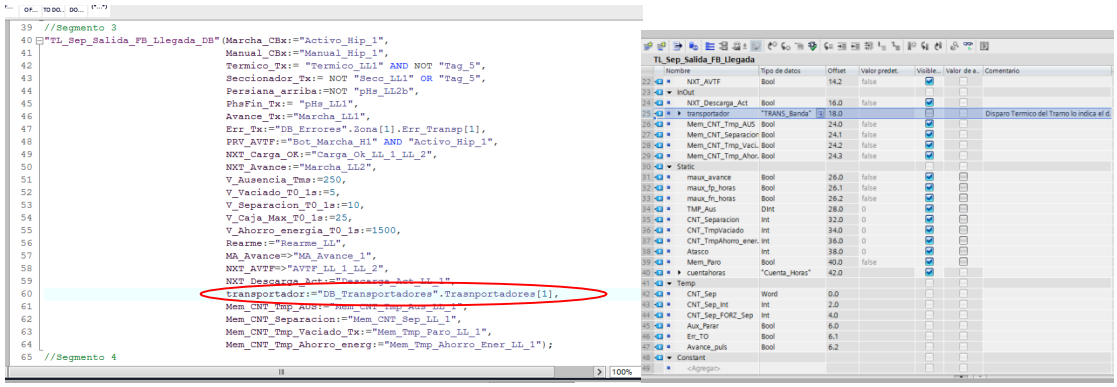


Ilustración 28: Como llamar a una estructura

Y dentro del bloque funcional TL_Sep_Salida_FB_Llegadas si se quiere acceder a las variables se hara de la siguiente manera:
transportador.Atasco:=TRUE;

CASO 5: Más de una llamada a un mismo DB

También otro caso crítico puede ser cuando se hace más de una llamada a un DB como es el caso de cuentahoras, esta variable es del tipo Cuenta_Horas, el cual es un FB. Por lo tanto, cuentahoras es una instancia del FB Cuenta_Horas y es como hacer una llamada al FB como se hace en el primer caso, aunque con una pequeña diferencia, y es que, esta vez como se declara como una variable local estática en vez de global aparece con una almohadilla y eso significa que cada vez que se la quiera llamar se debe declarar como estática, eso sí, si se quiere se puede declarar con el mismo nombre ya que no se pisan unas a otras.

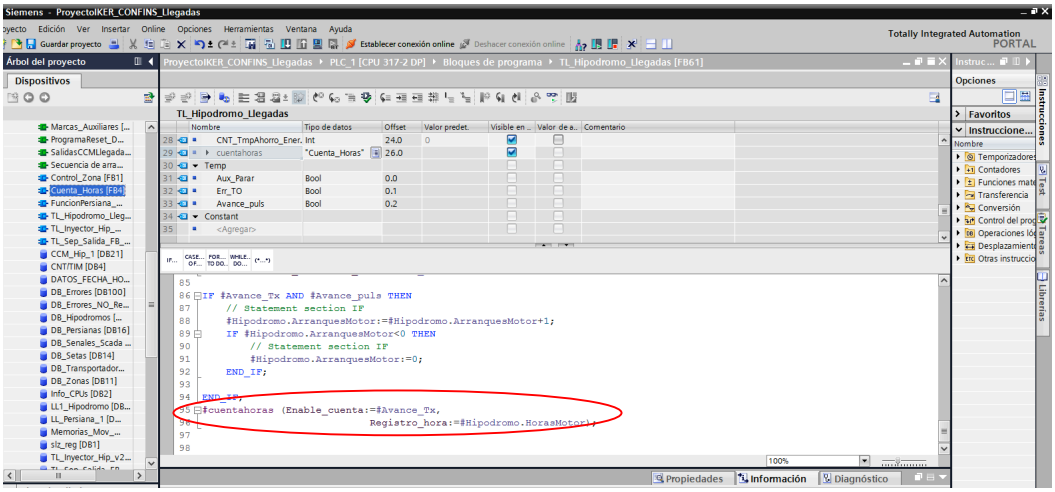


Ilustración 29: Llamadas Multiinstancia

CASO 6: Llamada a temporizadores TON

Hay veces que hace falta retardar ciertas variables antes de que sean activadas, por lo tanto, se llama a la función **TON.** Pero como este es un bloque funcional interno de Siemens se debe instanciarlo a un DB. Sería como en la siguiente imagen:

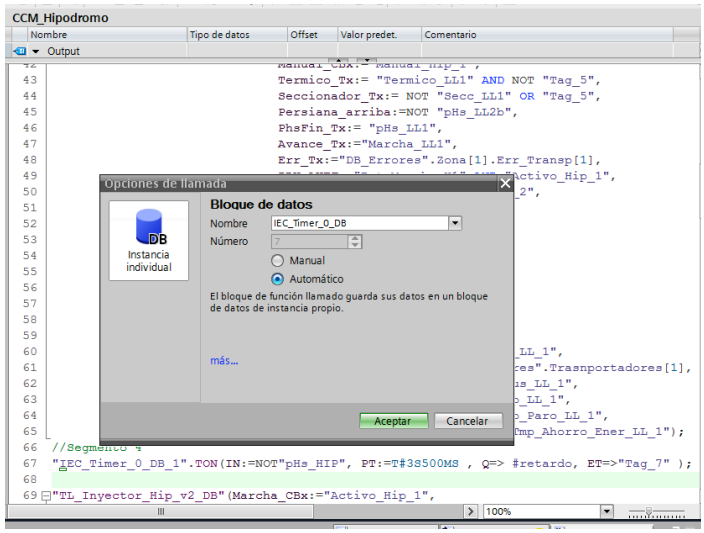


Ilustración 30: Bloque funcional TON

El tratamiento de este DB de instancia es como el tratamiento de otro DB de instancia de cualquier otro bloque funcional.

3.2.2.5.-SALIDAS CCM LLEGADAS (FC18)

En esta función se comprueba si se cumplen las condiciones para activar las salidas físicas, así se engloba en una función todas las activaciones y desactivaciones de los pines de salida. La programación de esta función se espera que se pueda aprovechar en otras plataformas tal y como está, ya que, solamente tenemos condiciones para activar las salidas.

3.2.2.6.-BORRADO ESTASDÍSTICAS (FC40)

La principal funcionalidad de esta función es la de resetear los diferentes DBs, por lo tanto, se llama a otra función para resetear y después se comprueba mediante variables globales si el borrado ha finalizado o no.

El único problema de esta función es que cuando se intenta borrar un DB de forma dinámica la forma de resetear dinámicamente no se hace de la misma manera en STEP7 y en TIA Portal. Por lo tanto, se tuvo que hacer una pequeña modificación del puntero ANY dentro de la función ProgramaReset_DB.

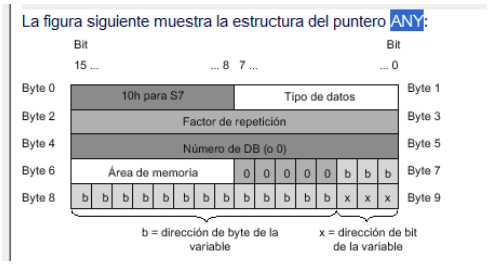


Ilustración 31: Declaración de un puntero ANY

La estructura del puntero Any y el rellenarlo de ceros quedaría de la siguiente manera:

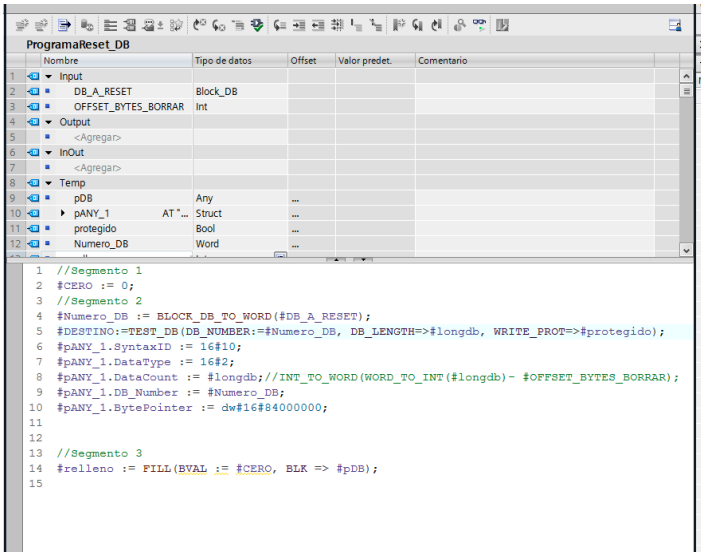


Ilustración 32: Programar un puntero ANY

3.2.2.7.-ERRORES HIPODROMO (FC431)

Esta función sirve para comprobar que tipo de error tiene mediante la generación de un código de error. En el caso de que hubiese un error y se pulsara rearme también nos sirve para borrar la variable código de error.

3.2.3.-SIMULADOR ULMA

Una vez traducido todo a SCL se decidió simular el programa en el simulador de ULMA de esta manera cerciorando que la traducción era correcta y no se iban a arrastras errores de programación en las siguientes migraciones. El siguiente pantallazo muestra como seria la instalación en el simulador.

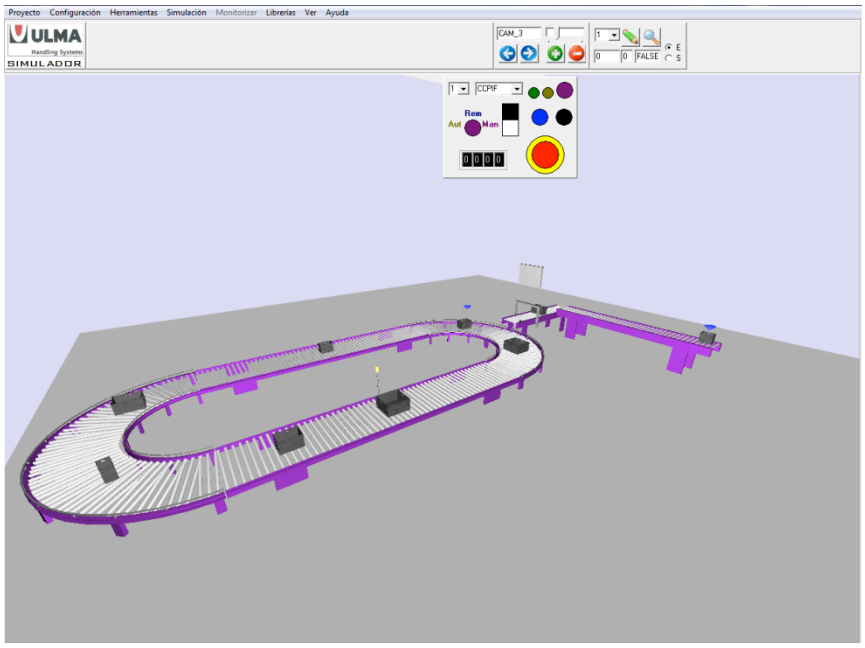


Ilustración 33: Simulador UHS

Para poder cargar en el simulador, previamente se debe definir el modelo del PLC al cual se volcara el programa. Pulsando sobre dispositivos y redes, cargará una pantalla parecida a esta.

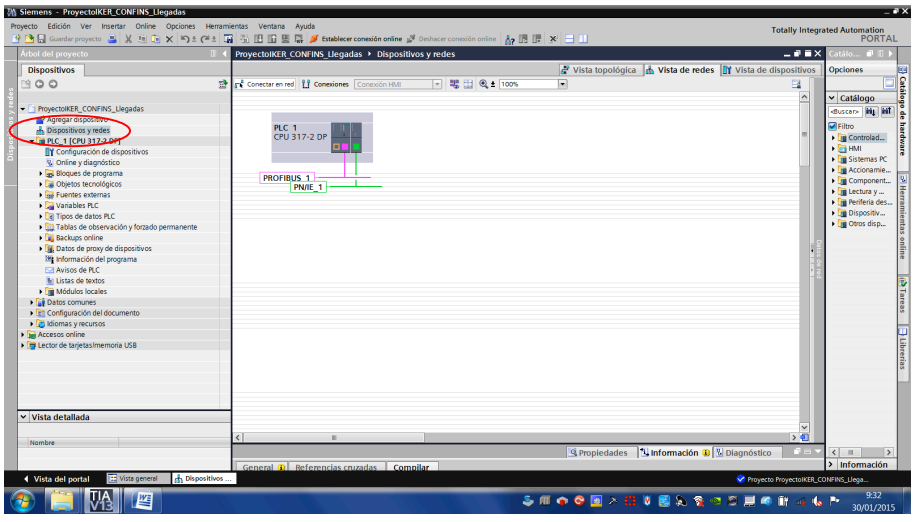


Ilustración 34: Ventana HW

En el caso de que el hardware cargado no corresponda con el modelo al que se quiera cargar, haciendo click derecho sobre el PLC aparecerá la opción de sustituir dispositivo.

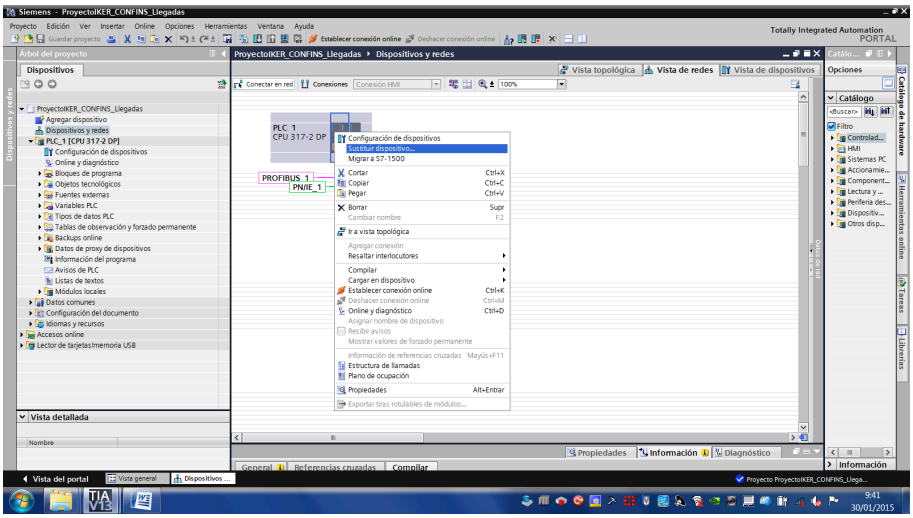


Ilustración 35: Como sustituir un modelo por otro

Una vez se tenga configurado el PLC correspondiente, se le asignará una IP para poder comunicar con el autómata. En este caso como este modelo de CPU no tiene integrado los puertos de conexión Ethernet, se tuvo que añadir un modulo de comunicaciones.

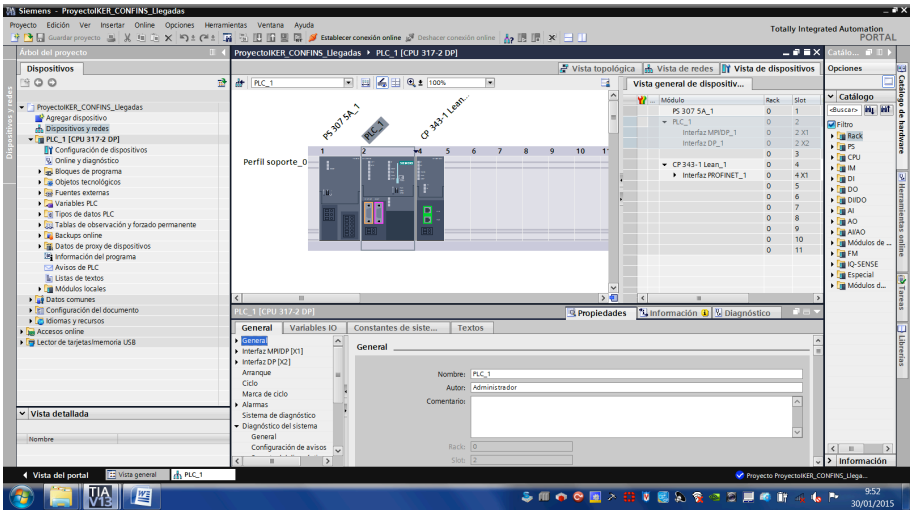


Ilustración 36: Configuración HW

Para finalizar con la comunicación se configurará el rango de IP que este libre para poder simular. Por ejemplo se puso:

IP autómata →	192.168.0.50
Máscara autómata →	255.255.255.0
IP simulador →	192.168.0.20
Máscara simulador →	255.255.255.0
IP portátil →	192.168.0.10
Máscara portátil →	255.255.255.0

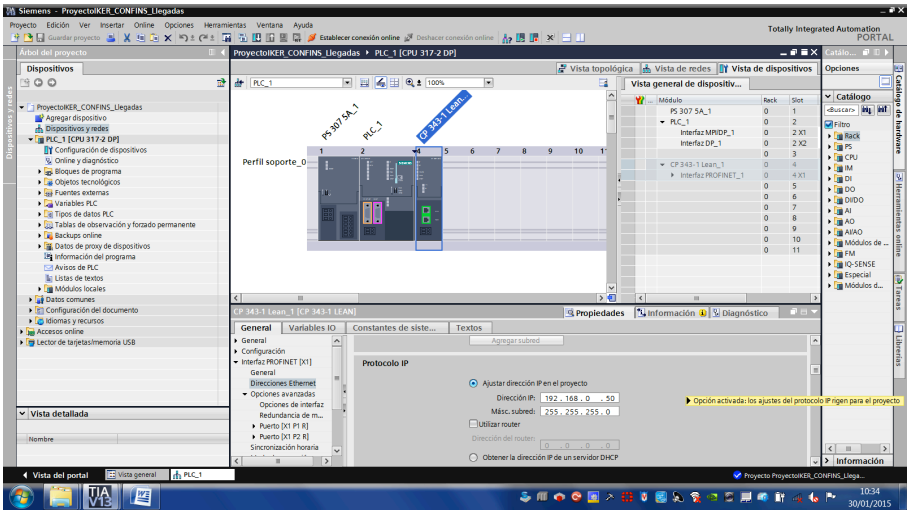


Ilustración 37: Configuración IP

Una vez este correctamente configurado el Hardware, se dispondrá a compilar y volcar el programa a un autómata. Para ello se seguirán los siguientes pasos:

PASO 1:

Una vez estén hechas correctamente todas las conexiones físicas con el autómata, se dispondrá a cargar el programa. Para ello, primero se clicará botón derecho sobre la carpeta donde está todo el programa del PLC, pulsando a su vez en cargar dispositivo. Aparecerán tres opciones, si es la primera vez que se vuelca el programa seleccionará la primera opción, Hardware y software (solo cambios).

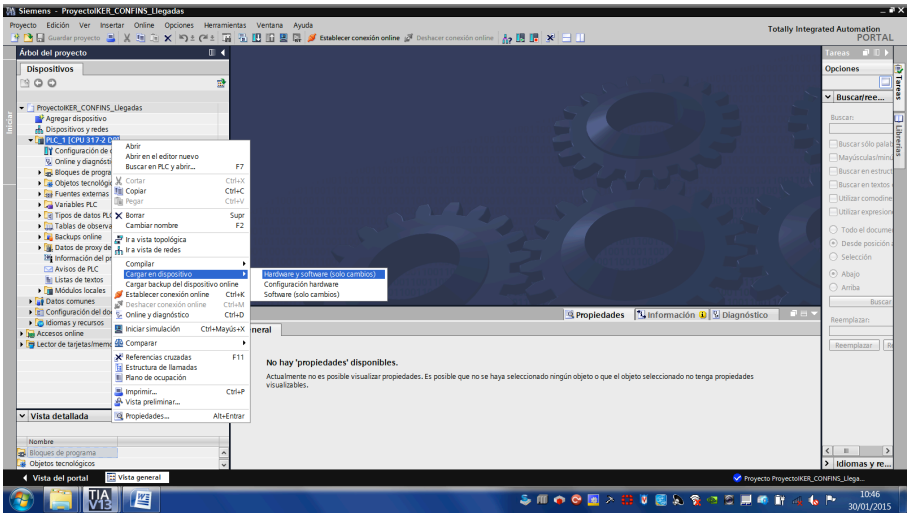


Ilustración 38: Cargar programa a PLC

Emergerá la siguiente ventana llamada “carga avanzada”, en esta pantalla se seleccionarán las opciones de la imagen y se pulsará sobre iniciar búsqueda.

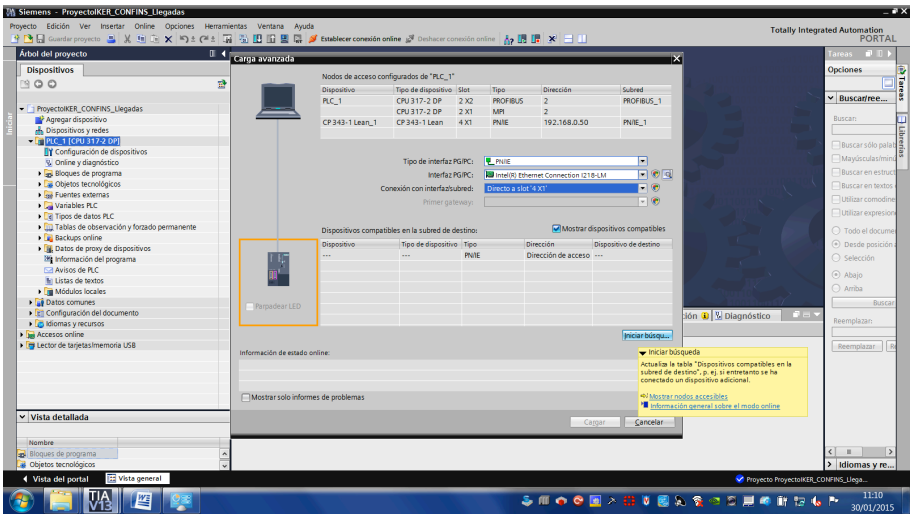


Ilustración 39: Búsqueda de elementos conectados

A continuación aparecerán los módulos que están conectados.

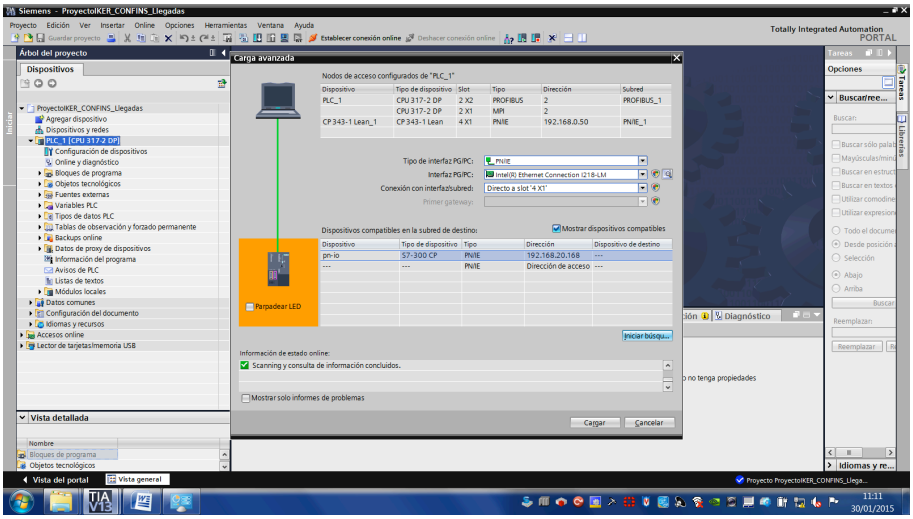


Ilustración 40: Dispositivos conectados

Si el rango del dispositivo es correcto continuar desde el **PASO 3**.

PASO 2:

Como se puede observar en la imagen anterior, la tarjeta de comunicación ya llevaba una dirección IP previamente configurada. Por lo tanto, a continuación, se explicarán los pasos que se deben seguir para fijarle nuestra IP.

Primero se buscará que dispositivos están accesibles y cuáles son sus direcciones IPs.

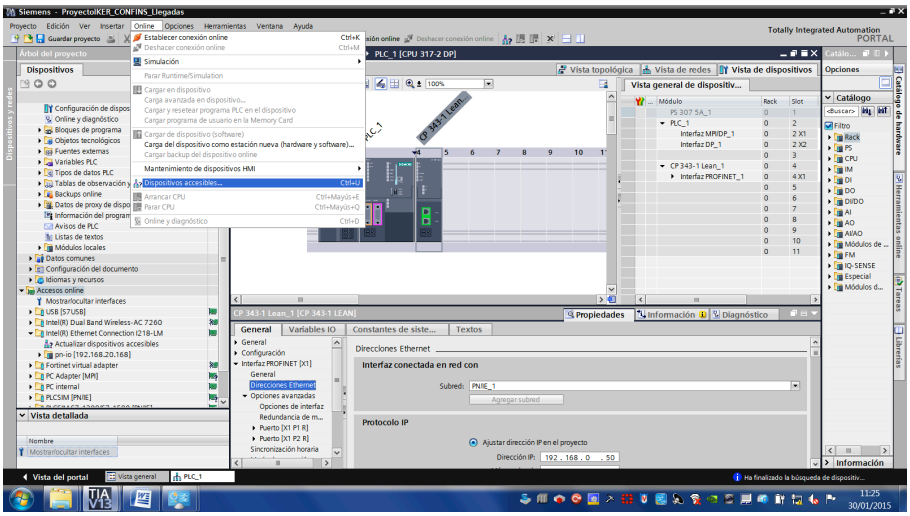


Ilustración 41: Búsqueda de dispositivos accesibles

Una vez se pulse automáticamente comenzará a buscar los dispositivos asociados, en cuanto finalice la búsqueda seleccione el modulo y pulse mostrar.

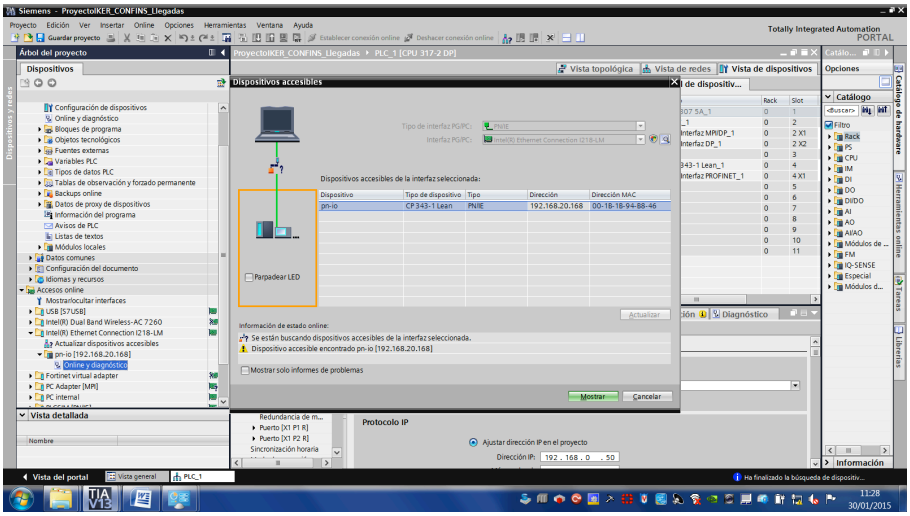


Ilustración 42: Mostrar los dispositivos accesibles

Acto seguido se abrirá el siguiente árbol, dentro de acceso online se seleccionara online y diagnostico como se puede observar en la imagen. De esta manera como se conecta con el dispositivo se podrá modificar la dirección.

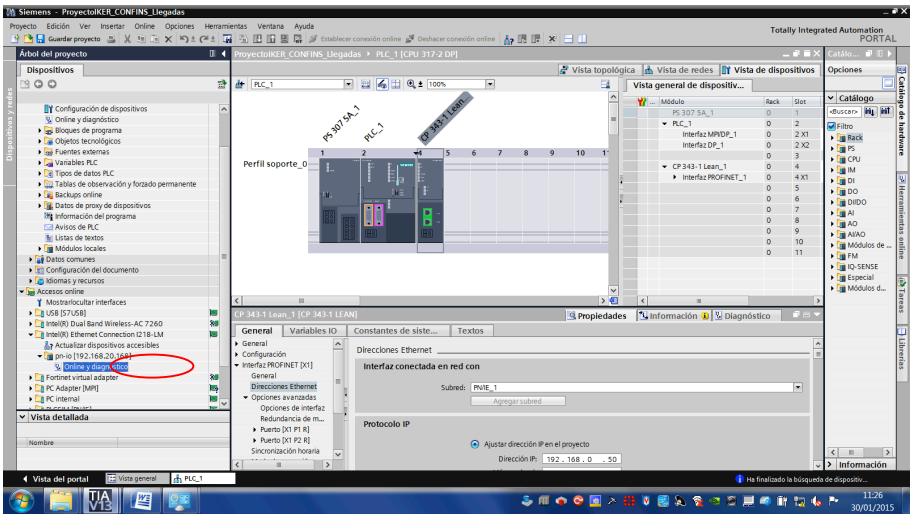


Ilustración 43: Conectarse al dispositivo

Una vez dentro de las características del dispositivo, seleccionando dentro de “funciones” “asignar direcciones” permitirá modificar la dirección IP del dispositivo.

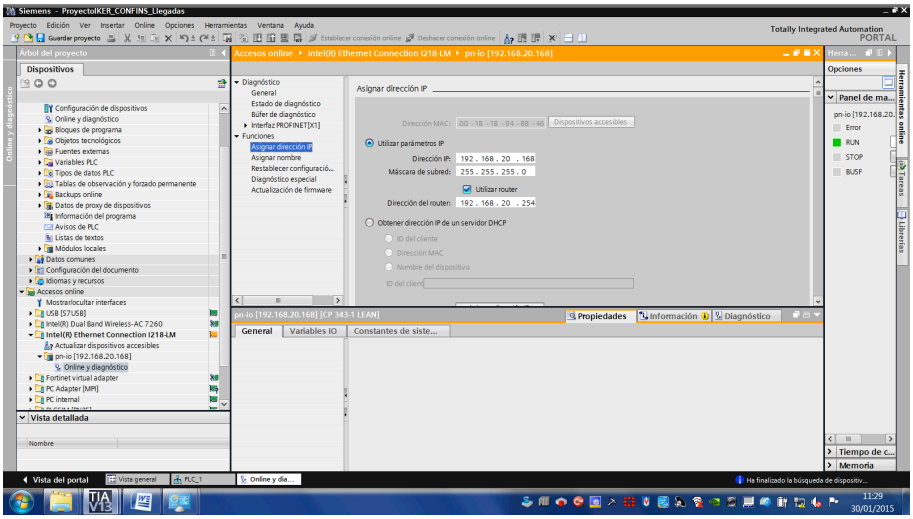


Ilustración 44: Cambiar IP del dispositivo

PASO 3:

Una vez esté configurado el dispositivo dentro del rango, se volverá a realizar el primer paso. Se cargarán en el dispositivo todo el Hardware y software.

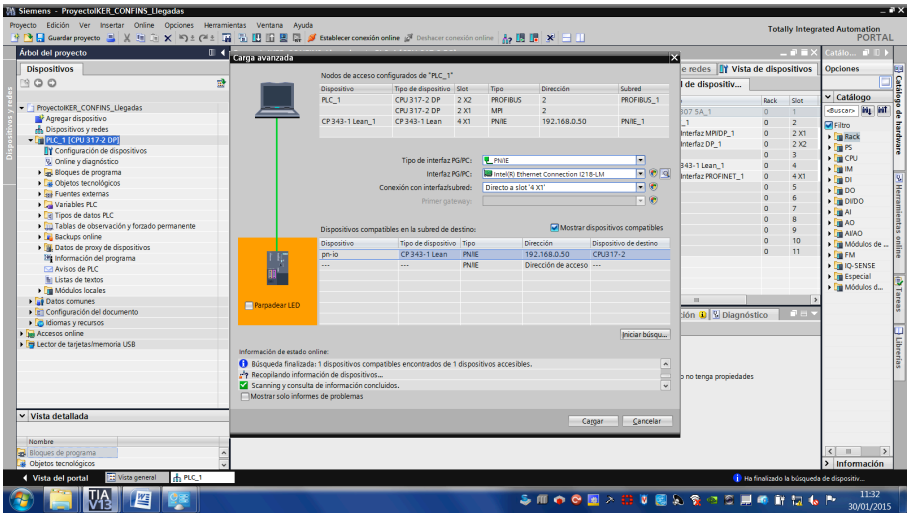


Ilustración 45: Búsqueda de elementos conectados

Esta vez el rango será correcto y el programa permitirá cargar el hardware y software al PLC. A la hora de cargar el programa, por seguridad, consultará algunas advertencias, pero seleccionando la opción de parar todo y aplicar todas se podrá cargar el programa.

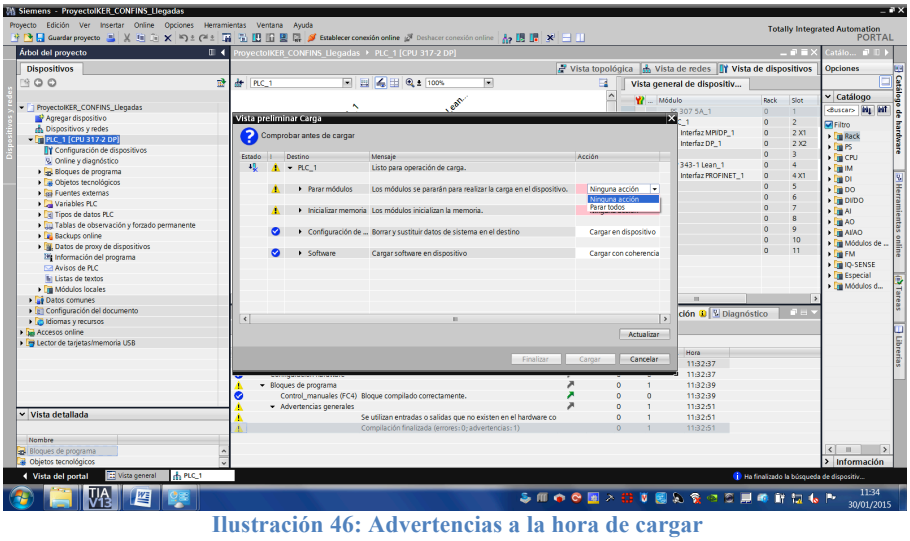


Ilustración 46: Advertencias a la hora de cargar

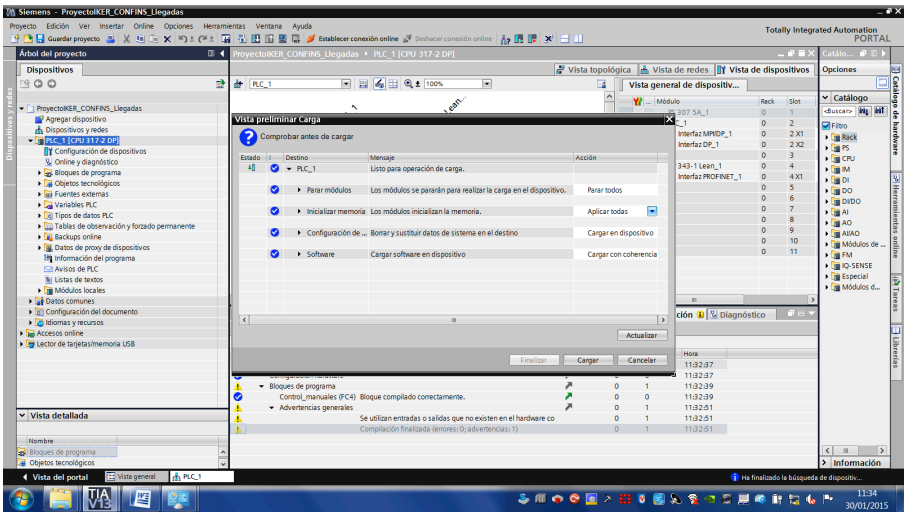


Ilustración 47: Advertencias a la hora de cargar

Una vez cargue todo aparecerá otra ventana, en el cual se pregunte si se quiere arrancar los módulos, se dejara como está por defecto y se pulsará finalizar.

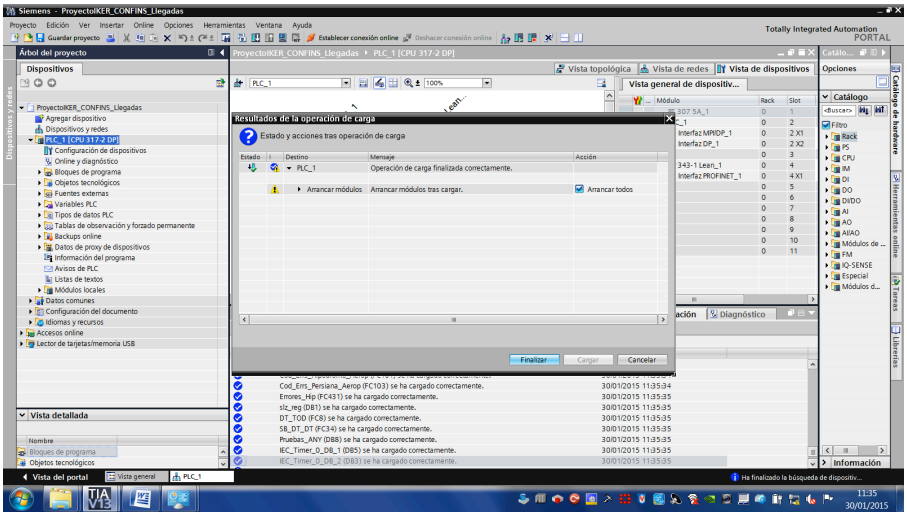


Ilustración 48: Hardware y software cargado

El siguiente paso será el de establecer la conexión. Si todo se ha hecho correctamente conectará sin problemas.

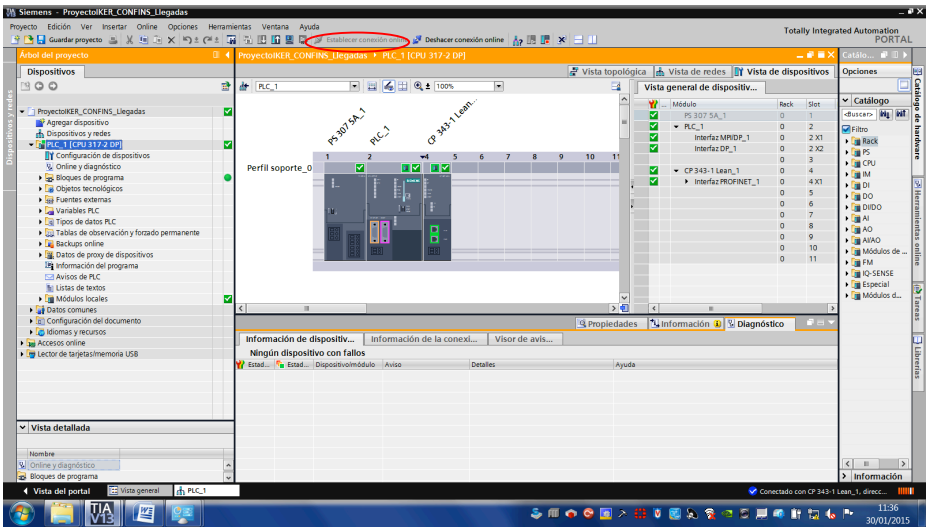


Ilustración 49: Conexión establecida

Por último, si se está simulando y se hace algún cambio en alguna función, el programa permite cargarlo en caliente, pulsando sobre la función botón derecho y seleccionando “cargar en dispositivo” “software (solo cambios)” se cargará solamente la parte modificada del programa.

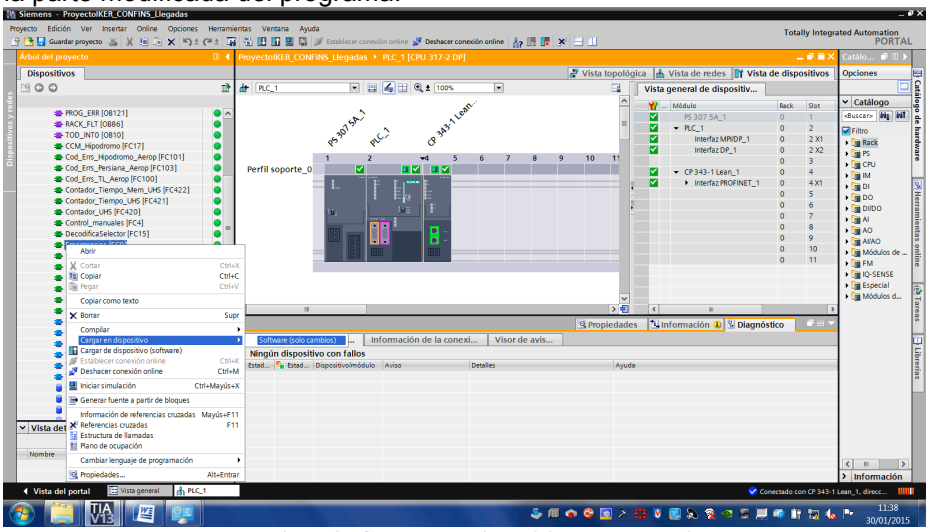


Ilustración 50: Cargar en dispositivo solo cambios

3.3.-MIGRACIÓN TIA PORTAL TWINCAT3

Finalizada la traducción del programa a SCL, llego el momento de comprobar cuales eran las diferencias entre Siemens y el estándar IEC61131-3. Para llevar a cabo esta fase se instaló el programa Twincat3 de Beckhoff (para cualquier consulta de instalación véase el manual de instalador).

3.3.1.- OPCIONES PARA MIGRAR DE UNA PLATAFORMA A OTRA

a) En un principio se optó por buscar la vía rápida de la exportación e importación, pero se vio que ni uno tenía la opción de exportar a PLCopenXML ni el otro tenía la opción de importar de Siemens. Por lo tanto, como cada uno tiene una opción de exportación diferente queda anulada esta opción de traducción directa.

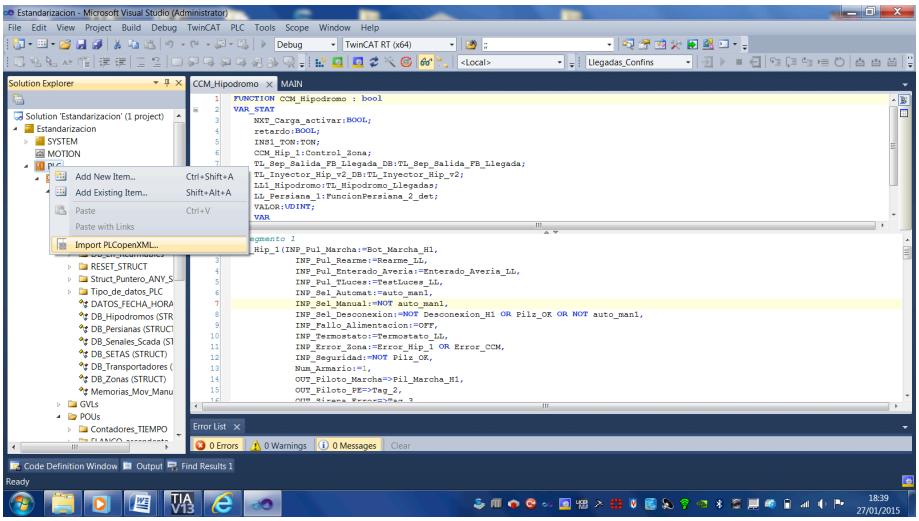


Ilustración 51: Exportación mediante PLCopenXML

b) Otra solución podía ser la de copia y pega, en este caso los resultados fueron muy favorables pero se puede decir que funciona mejor en una dirección que en la otra. Esto es, si se traduce el programa escrito en Siemens a Beckhoff no es tan directa como si se hiciese al revés. En la siguiente imagen se puede ver mejor como los dos códigos no son del todo iguales.

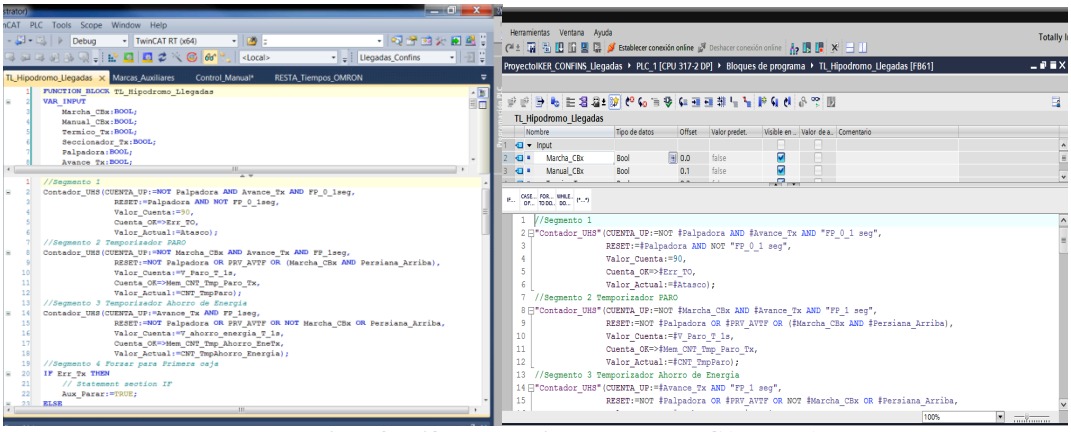


Ilustración 52: Diferente código entre TWINCAT y TIA

En este caso como el programa está en TIA Portal, los pasos a seguir para la traducción fueron los siguientes:

PASO 1:

Crear la función dentro de Unidades de Organización de Programa (POU)

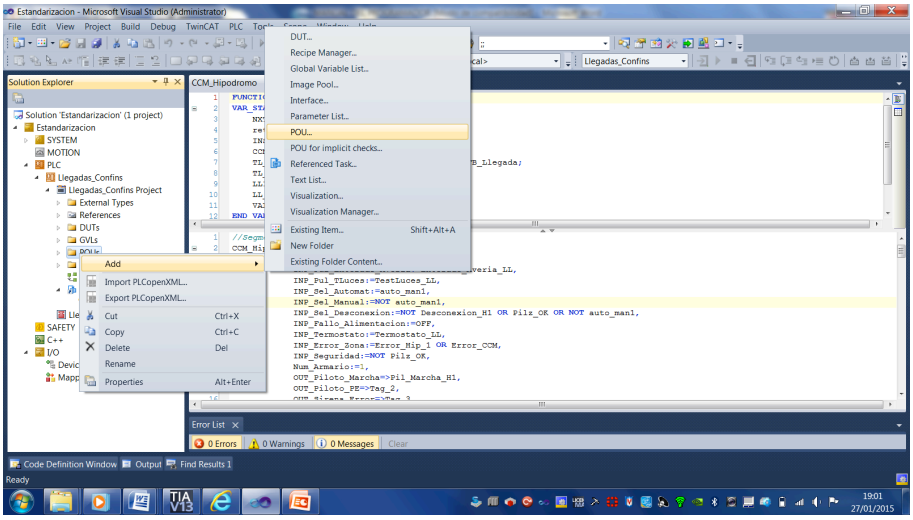


Ilustración 53: Crear funciones en Twincat3

Acto seguido aparecerá la siguiente ventana donde se definirán el nombre, el tipo de POU y en que lenguaje se quiere escribir el programa.

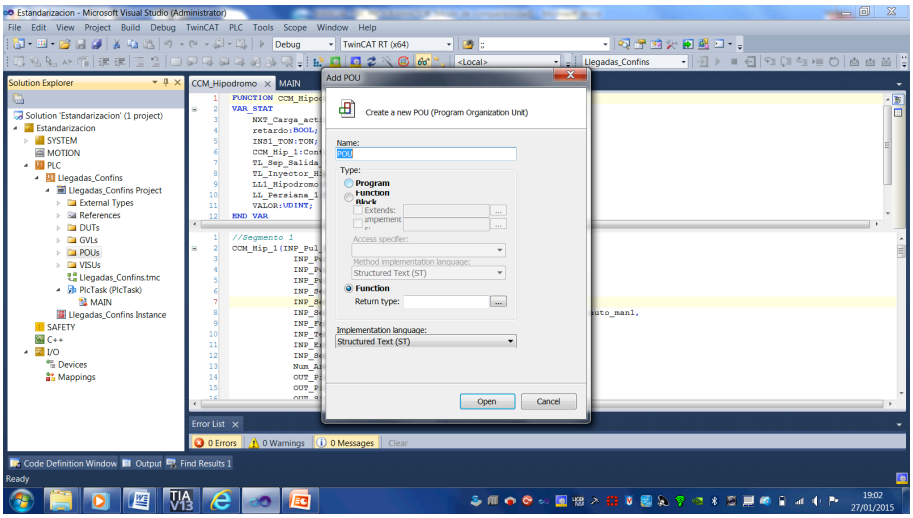


Ilustración 54: Seleccionar tipo de función y lenguaje

PASO 2:

Una vez se tenga creada la función se pegará el código de la parte de SIEMENS. Pero como se mencionó anteriormente tenemos más caracteres de los necesarios, por lo tanto se sustituirán aquellos caracteres por nada. Para ello primero se pulsa CTRL+F para buscar y después se escoge la opción de reemplazar.

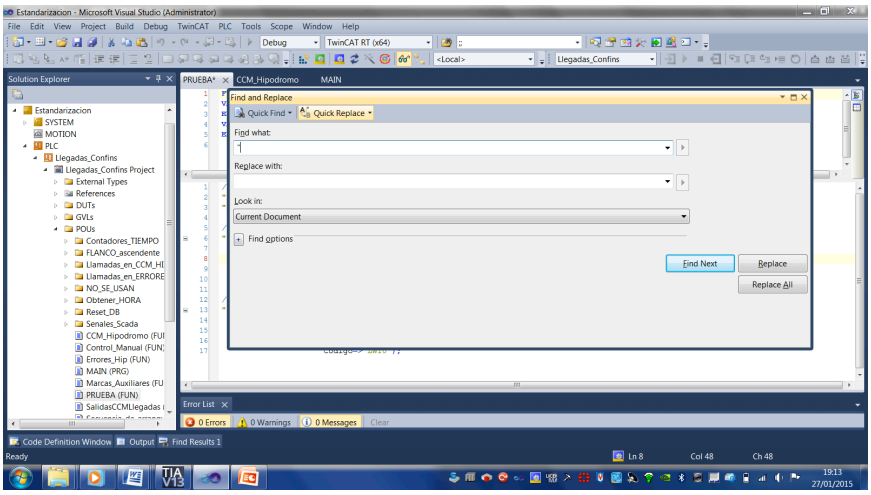


Ilustración 55: Búsqueda de caracteres para ser sustituidas

PASO 3:

También se introducirán las declaraciones de las variables copiando y pegando. Pero, en este caso como en una plataforma se divide la declaración en celdas y en la otra no se divide, la declaración en la plataforma Twincat3 no sería correcta.

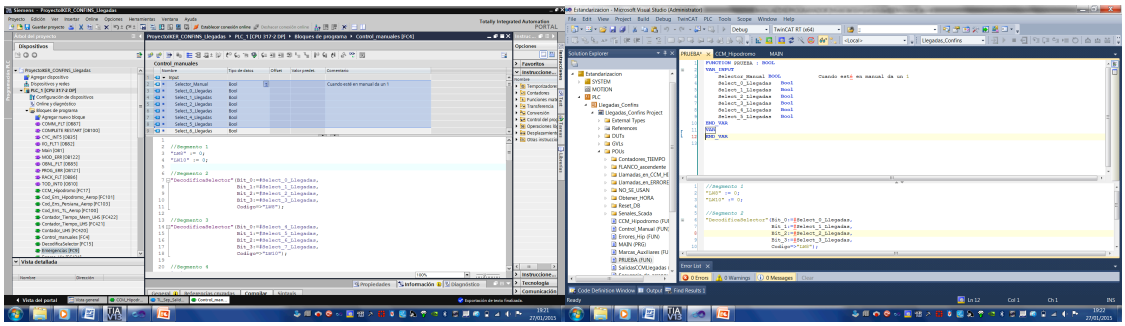


Ilustración 56: Diferencia en la declaración de variables

Para que la declaración fuese la correcta se tuvo que añadir después de cada nombre de variable ":" y al final del tipo ";". En la siguiente imagen se ve mejor.

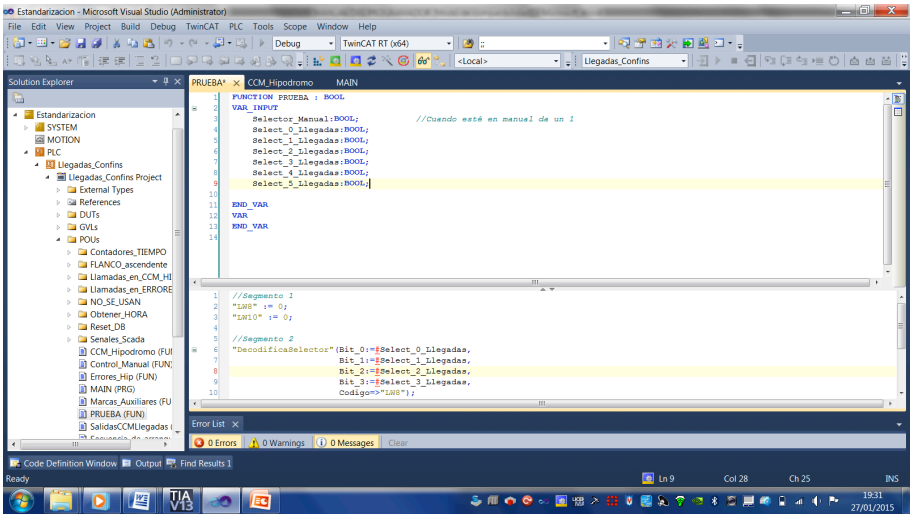


Ilustración 57: Declaración de variables en Twincat3

Y este proceso se debe repetir con todas las funciones.

3.3.1.- INCOMPATIBILIDADES ENTRE LAS DOS PLATAFORMAS

Una vez estudiada la mejor forma de trasladar el código de una plataforma a otra, se tradujo todo lo programado y verificado en Tia Portal a Twincat3. Pero esta traducción no fue tan directa como se esperaba, por lo tanto se tuvo que hacer algunas pequeñas modificaciones.

Como la organización y estructuración del programa era igual que en TIA Portal, a continuación se enumeraran las modificaciones más relevantes:

a) La primera modificación fue la de crear estructuras en vez de DBs, ya que, la filosofía de Data Blocks solamente la tiene Siemens todas las demás marcas que siguen el estándar y lo hacen mediante estructuras. La idea es la misma, solamente que en vez de saber la dirección en la que se encuentra la variable sobre la que se quiera actuar (p.ej. DB1.DBX1.0), ahora se debe llamarlo mediante una instancia a la estructura sobre la que se encuentra (p.ej. INS_DB_Persianas.Persianas[1].Bajar). Declarándolos como estructuras nos puede facilitar la programación en algunos casos.

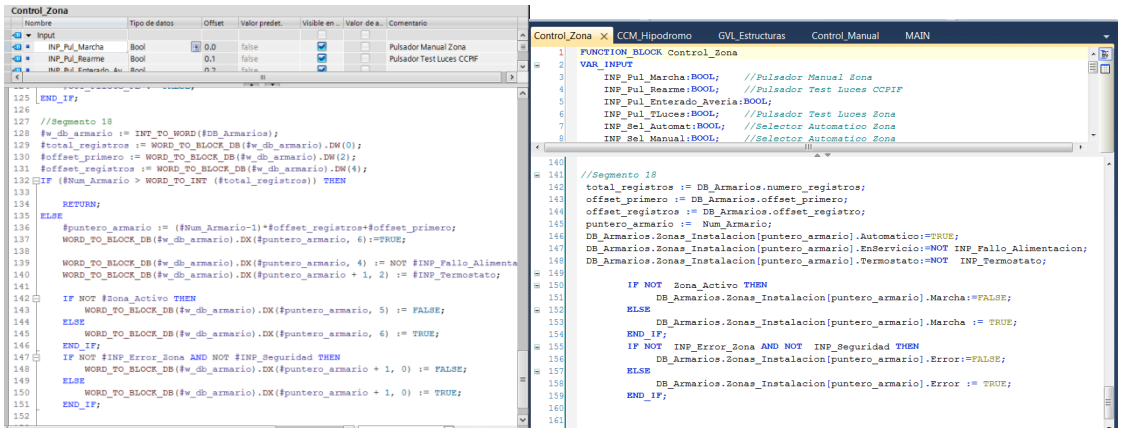


Ilustración 58: Ventajas al acceder a las variables de una estructura

En esta imagen se puede ver dos casos diferentes donde se llama al mismo DB o estructura, la imagen de la izquierda muestra como se puede hacer una llamada indirecta a un DB. En cambio, en la imagen de la derecha se puede observar como se llama a la estructura DB_Armarios, que a su vez es una instancia de DB_Zonas. Pero en este caso como habia más de un Armario se creó un Array de DB_Zonas y así mantenía la opción de llamar a la zona que se quisiera.

b) Modificación en la función ProgramaReset_DB, la principal funcionalidad de esta función, como se explicó anteriormente, es la de resetear cualquier tipo de estructura o DB. Pero en Siemens y en Beckhoff es totalmente diferente la manera de hacer un puntero Any. En la siguiente imagen se ejemplifica lo mencionado.

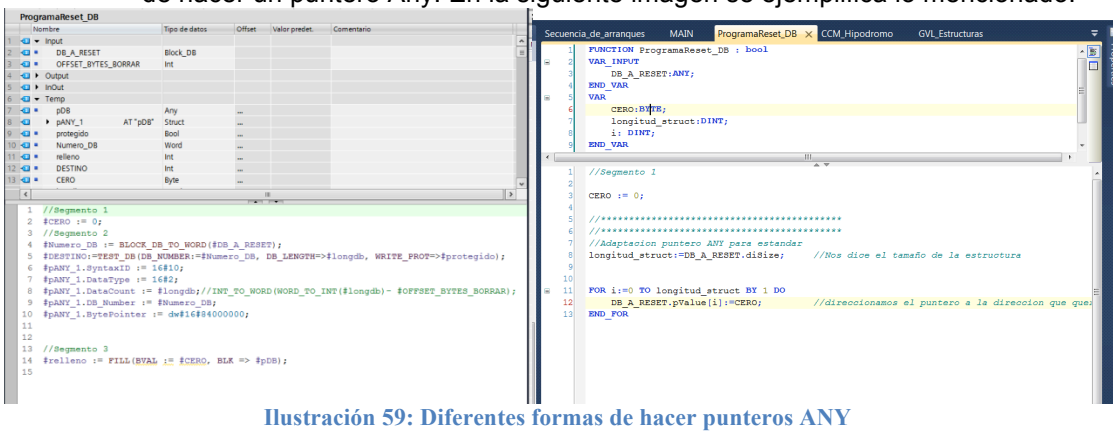


Ilustración 59: Diferentes formas de hacer punteros ANY

Como se observa, la manera de hacer un puntero Any en Beckhoff es mucho más sencillo que en Siemens, ya que el estándar no nos obliga a crearnos una estructura de tipo Any como lo hace Siemens. Y además para resetear la estructura nos es suficiente con recorrer toda la estructura y ponerla a cero, de esta manera evitando funciones internas como FILL para el caso de Siemens.

c) Otra diferencia con Siemens es que cuando hacemos una instancia a un FB, Siemens cada vez que crea un FB obliga a crear un DB de instancia. En cambio Beckhoff no te indica que se debe instanciar el FB, pero para que una FB pueda ser llamada por el programa previamente debe ser declarada, bien como global o local. En este caso, se han declarado todos como locales estáticas, de esta manera no se pierden los datos cada vez que se deja de llamar a la función.

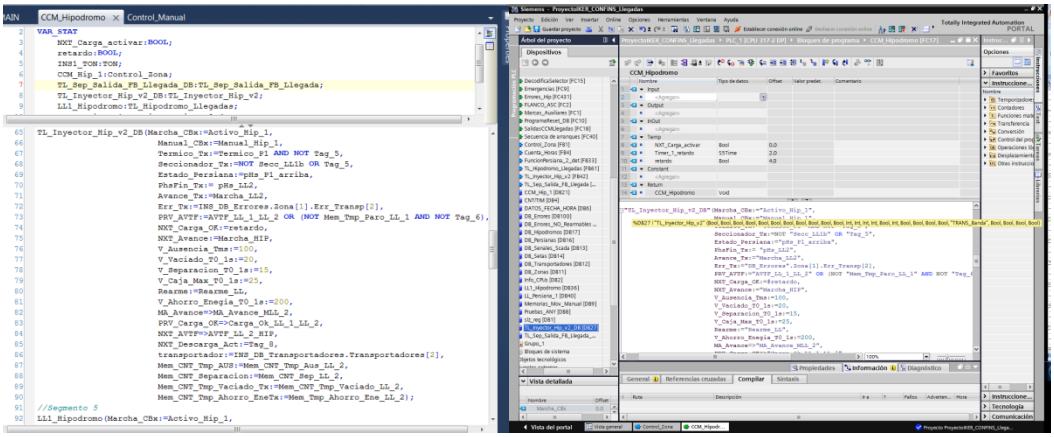


Ilustración 60: Diferencias en instanciar una FB

También debemos declarar como globales todas las instancias de aquellas estructuras que serán llamadas desde diferentes puntos del programa. Porque lo demás perderíamos información y dejarían de funcionar como estructuras, y funcionarían como variables locales.

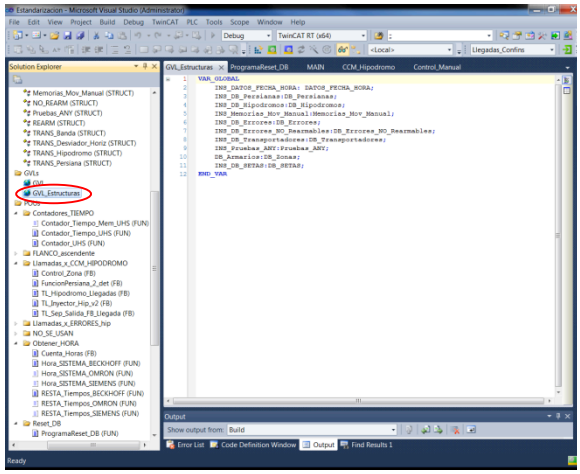


Ilustración 61: Declaración global de estructuras

d) La quinta diferencia puede que sea la más llamativa, puesto que hace referencia a la obtención de la fecha y hora. Como cada plataforma tiene su propia función para obtener la hora del sistema, se decidió crear diferentes funciones para que así cada plataforma llamase a su función. Pero, como cada función devuelve un resultado de un tipo concreto, se debe transformarlo a un tipo que sea común para todas las funciones y así evitar modificar el programa.

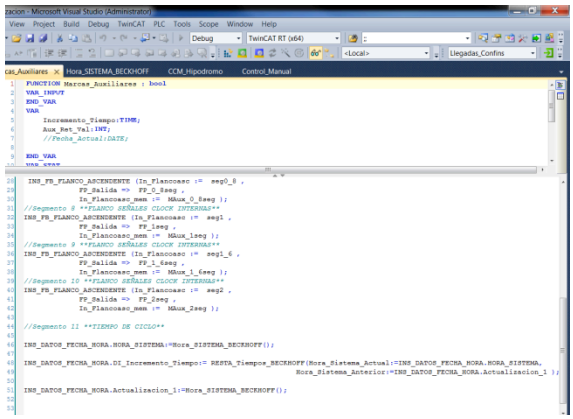


Ilustración 62: Cálculo con variables del tipo hora

Por ejemplo, en Beckhoff la función se llama GETSYSTEMTIME y devuelve un valor tipo UDINT pero se debe convertirlo a Date And Time porque es la única manera de que las dos plataformas nos devuelvan un valor del mismo tipo.

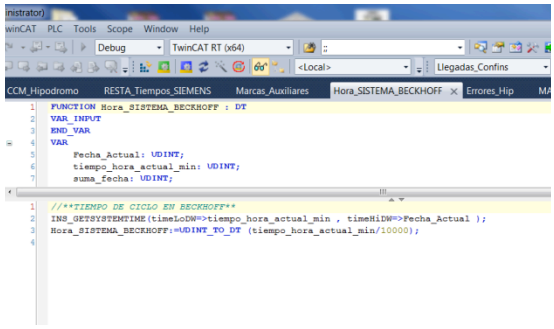


Ilustración 63: Función para obtener la hora

En Siemens en cambio, la función se llama RD_SYS_T, pero el valor que devuelve es del tipo Date And Time por lo tanto no se debe convertir nada.

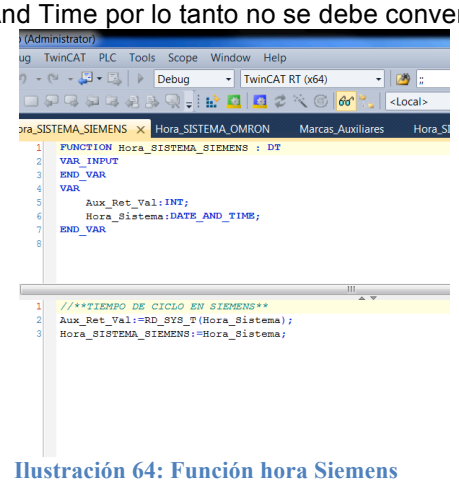


Ilustración 64: Función hora Siemens

Además de las funciónes para la obtención de la hora en cada plataforma, se han creado otras para restar horas en cada programa. Como en Siemens no se puede convertir la hora a UDINT y restar simplemente, se ha tomado la decisión de crear diferentes funciones para cada plataforma y de esta manera llamamos a aquella función que nos calcula la resta entre tiempos. En la siguiente imagen se muestran las dos funciones.

BECKHOFF SIEMENS

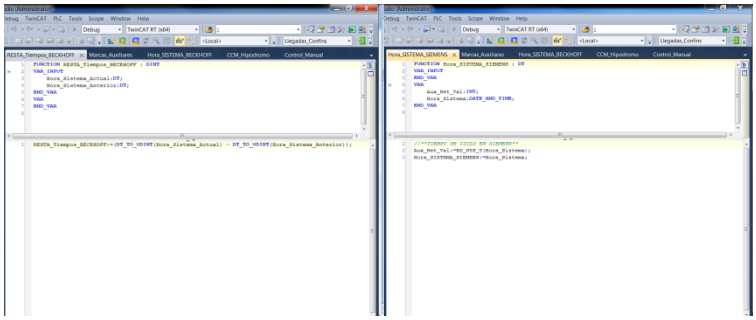


Ilustración 65: Funciones para hacer cálculos con horas

e) Y para finalizar modificamos la manera de crear variables cíclicas. Como en el estándar no tenemos ningún tipo de función u opción, las hemos tenido que crear mediante retardos y pienso que esta opción es más que interesante, puesto que todas las plataformas admiten este tipo de retardos puede que sea la forma más común de hacer este tipo de llamadas cíclicas. Además de poder configurar la latencia a nuestro antojo.

```
10    INS_TON_0_1s(IN:= NOT INS_TON_0_1s.Q, PT:=T#0.1S);
```

Ilustración 66: Variables cíclicas en Beckhoff

3.4.-EXPORTAR E IMPORTAR desde BECKHOFF/CODESYS a OMRON

Para dicha prueba instale el software mas reciente que tiene OMRON en el mercado, Sysmac Studio v1.0. Y se comprobó si realmente OMRON cumplía con el estándar IEC61131 al completo y si se podía exportar todo fácilmente con un solo clic.

3.3.1.- COMO MIGRAR DE UNA PLATAFORMA ESTÁNDAR A SYSMAC STUDIO

Pero no fue tan sencillo. OMRON solo admite importar y exportar archivos de su propia plataforma como se puede observar en la foto.

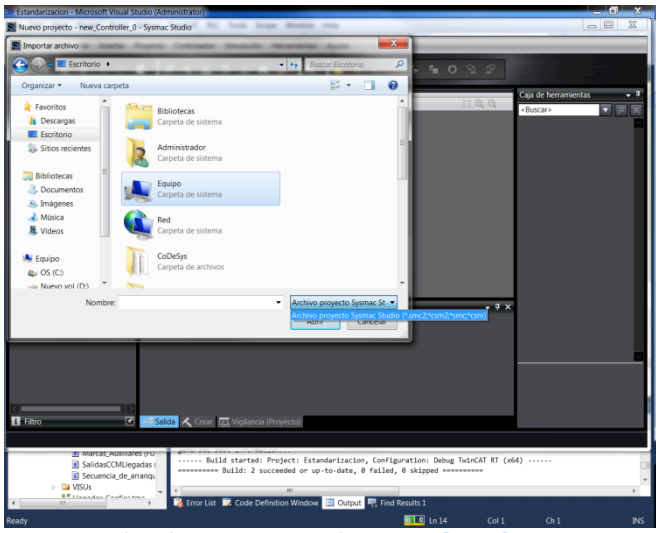


Ilustración 67: Formas para importar OMRON

Lo ideal hubiese sido que OMRON permitiese exportar e importar archivos en formato PLCopenXML.xml, ya que guardando en formato ".xml" se podría migrar archivos desde Codesys a OMRON o viceversa. Por lo tanto, la técnica de migración será la misma que se empleó para Siemens, esto es, la de copiar y pegar el código.

PASO 1:

Se crearán primero todas las funciones y bloques funcionales, como se hizo previamente en la plataforma Twincat3.

PASO 2:

Una vez esté la estructura del programa se copiará y pegará el código correspondiente. Pero en vez de copiar el código del TIA Portal se copiará el del Twincat3, ya que, el método de escritura de twincat3 es compatible con el de Sysmac Studio.

PASO 3:

La declaración de variables será la parte más costosa de migrar, ya que, la única forma directa de hacerlo sería creando un documento Excel con la misma forma que la tabla de variables, así copiando y pegando las columnas ya podríamos añadirlas. Aunque exista una forma directa de copiar estas variables es menos costoso escribir el nombre y tipo de la variable de uno en uno.

3.3.1.- INCOMPATIBILIDADES ENTRE EL ESTÁNDAR Y SYSMAC STUDIO

Una vez copiado todo el programa en la nueva plataforma, se compilará el programa para ver las diferentes incompatibilidades entre el código de Twincat3, el cual está basado en el estándar IEC61131, y el código escrito en la plataforma Sysmac Studio.

a) La primera incompatibilidad fue la manera de obtener la Fecha y Hora, como se mencionó anteriormente cada plataforma tiene su propia función para la hora. Por lo tanto, si se quiere tener los contadores de tiempos en función del tiempo transcurrido, se debe crear otra función para el cálculo de la hora en Sysmac Studio.

   Como se hizo para Siemens y Beckhoff primero se debe obtener la hora y después se calculará la diferencia de tiempos mediantes restas.

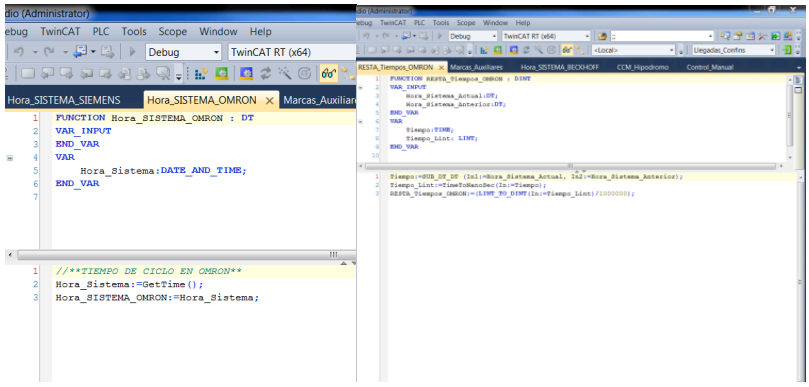


Ilustración 68: Funciones para obtener y calcular la hora en OMRON

b) También las llamadas a las variables cíclicas en cada plataforma son diferentes. Por ejemplo, en Siemens se definen en el hardware y es un byte que está compuesto por 7 bits, donde cada bit se activa a una frecuencia diferente. En la siguiente imagen se puede observar mejor las latencias:

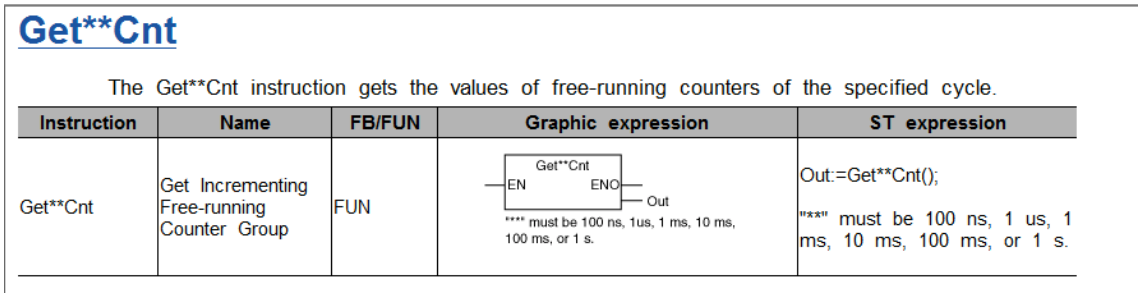
Duración del período

Cada bit del byte de marcas de ciclo lleva asignada una duración de período/frecuencia:

Bit	7	6	5	4	3	2	1	0
Duración del período (s):	2	1,6	1	0,8	0,5	0,4	0,2	0,1
Frecuencia (Hz):	0,5	0,625	1	1,25	2	2,5	5	10

Ilustración 69: Variables cíclicas SIEMENS

En OMRON, en cambio, las llamadas a las variables cíclicas se hacen mediante una función específica:



Get**Cnt

The Get**Cnt instruction gets the values of free-running counters of the specified cycle.

Instruction	Name	FB/FUN	Graphic expression	ST expression
Get**Cnt	Get Incrementing Free-running Counter Group	FUN	Get**Cnt EN ENO Out "**" must be 100 ns, 1us, 1 ms, 10 ms, 100 ms, or 1 s.	Out:=Get**Cnt(); "**" must be 100 ns, 1 us, 1 ms, 10 ms, 100 ms, or 1 s.

Ilustración 70: Variables cíclicas OMRON

c) Por último, mencionar, que en OMRON no existen variables de tipo ANY. Por lo tanto, el reset de las estructuras se debe hacer mediante un Array de estructuras, siempre y cuando se quiera que el reset sea de forma indirecta. Lo demás llamando directamente a la estructura que se deseé borrar es suficiente.